Simulation Acceleration
Algorithm Acceleration
Logic Emulation
ASIC Emulation
ASIC Verification
FPGA Boards
Consulting

User Guide

DN9200K10PCIE8T

DN9200K10PCIE8T User Manual
Major Revision 1

Last Update March 27, 2009 by fullsail user

7469 Draper Avenue
La Jolla, CA92037 USA
Phone 858.454.3419 • Fax 858.454.1728
support@dinigroup.com
www.dinigroup.com

---

1 Table of Contents

1 TABLE OF CONTENTS....5

2 LIST OF FIGURES....15

CHAPTER 1: INTRODUCTION....19

1 MANUAL CONTENTS....19

1.1 INTRODUCTION....19
1.2 QUICK START GUIDE....19
1.3 CONTROLLER SOFTWARE....20
1.4 HARDWARE....20
1.5 THE REFERENCE DESIGN....20
1.6 ORDERING INFORMATION....20

2 AUDIENCE....20

3 CONVENTIONS....20

3.1 NOTATIONS....21
3.2 FILE PATHS....21
3.3 PHYSICAL DIMENSIONS....21
3.4 PART PIN NAMES....21
3.5 SCHEMATIC CLIPPINGS....21

4 GLOSSARY....22

5 RESOURCES....24

5.1 USER CD....24
5.2 DINIGROUP.COM....26
5.3 ERRATA LIST....26
5.3.1 Existing Errata....26
5.4 REFERENCE DESIGN....26
5.5 SCHEMATICS AND NETLIST....26
5.5.1 Netlist....26
5.5.2 Net name conventions....27
5.6 DATASHEET LIBRARY....27
5.7 XILINX....28
5.8 DINI GROUP REFERENCE DESIGNS....28
5.9 BOARD MODELS....28
5.9.1 Base System Builder....28
5.9.2 Using Partitioning and 3rd party synthesis tools....28
5.10 PCI EXPRESS DETAILS....28

5.11 EMAIL AND PHONE SUPPORT 28
CHAPTER 2: QUICK START GUIDE 31
1 PROVIDED MATERIALS 31
1.1 SYSTEM REQUIREMENTS 32
2 WARNINGS 32
2.1 ESD 32
2.2 OTHER 33
3 PRE-POWER ON INSTRUCTIONS 33
3.1 INSTALL MEMORY (OPTIONAL) 34
3.2 PREPARE CONFIGURATION FILES 34
3.3 INSERT THE COMPACT FLASH CARD 34
3.4 INSTALL DN9200K10PCIE8T IN COMPUTER (OPTIONAL) 34
3.5 CONNECT RS232 CABLE 35
3.6 CONNECT USB CABLE 35
3.7 CONNECT POWER CABLE 35
3.8 DAUGHTER CARDS 36
4 POWER ON INSTRUCTIONS 36
4.1 VIEW CONFIGURATION FEEDBACK OVER RS232 36
4.2 CHECK LED STATUS LIGHTS 38
5 RUN USB CONTROLLER 39
5.1 DRIVER INSTALLATION 39
5.2 OPERATING THE USB CONTROLLER PROGRAM 40
5.2.1 Configure an FPGA 41
5.2.2 Set Clock Frequencies 42
5.3 RUN HARDWARE TESTS 42
5.3.1 Clock Frequencies 42
5.3.2 DDR2 42
5.3.3 Other Hardware Tests 43
5.4 GETTING DATA TO AND FROM THE FPGA 43
6 RUN AETEST_WDM 44
6.1.1 Use AETest 44
7 SCAN THE JTAG CHAIN 46
8 MOVING ON 47
CHAPTER 3: CONTROLLER SOFTWARE 49
1 USB CONTROLLER 50

1.1 MAIN WINDOW 50
1.1.1 Refresh Button *51*
1.1.2 Disable/Enable USB *51*
1.1.3 Log Window *52*
1.1.4 Board Graphic *52*
1.2 MENU OPTIONS 53
1.2.1 File Menu *53*
1.2.2 Edit Menu *54*
1.2.3 FPGA Configuration Menu *54*
1.2.4 FPGA Reference Design *55*
1.2.5 Main Bus *55*
1.2.6 Settings/Info Menu *56*
1.2.7 Production Test *57*
1.2.8 Service Menu *58*
1.2.9 Debugging Menu *58*
1.3 INI FILE 58

2 AETEST USB 58

3 PCI EXPRESS AETEST APPLICATION 58

3.1 COMPILING AETEST_USB 59
3.1.1 Compiling the Driver *59*
3.2 FUNCTIONALITY 59
3.3 RUNNING AETEST 60

4 ROLLING YOUR OWN SOFTWARE 62

4.1 USB 62
4.1.1 Windows XP/Vista *62*
4.1.2 Linux *62*
4.2 PCIE 63
4.2.1 Windows Driver Hooks *63*
4.2.2 Linux Driver Hooks *64*

5 UPDATING THE FIRMWARE 64

5.1 OBTAINING THE UPDATES 65
5.2 UPDATING THE SPARTAN (PROM) FIRMWARE 65
5.2.1 Using JTAG cable *65*
5.2.2 Using USBController *67*
5.2.3 Using AEtest_USB *68*
5.3 UPDATING THE MCU (FLASH) FIRMWARE 69
5.4 PCI EXPRESS ENDPOINT FIRMWARE 70
5.4.1 Using JTAG USB cable (Xilinx products - iMpact) *70*
5.4.2 Using USBController *72*
5.4.3 Using AETest_USB *72*

CHAPTER 4: HARDWARE 73

1 GENERAL OVERVIEW 73

2 VIRTEX 5 FPGAS 74

2.1 STUFFING OPTIONS 74
2.1.1 Q: So Can I get two SX240s? 74
2.1.2 FPGA A and B: 74
2.1.3 CES Parts 74
2.1.4 "Small" FPGAs 74
2.1.5 FPGA Q (PCI Express FPGA) Options 76
2.1.6 Speed Grades 77
2.2 USING IO 77
2.2.1 Timing 77
2.3 HARDWARE ERRATA DETAILS 78
2.4 UPGRADE POLICY 78
2.4.1 Upgrading to new board 78
2.4.2 Adding FPGAs to a DN9200K10PCIE8T 78

3 PCB 78

3.1 TRACE DELAY 78
3.2 SIGNAL QUALITY 78

4 CONFIGURATION SECTION 78

4.1 CONFIGURATION SECTION FEEDBACK 79
4.2 FPGA CONFIGURATION 80
4.3 PCI EXPRESS 82
4.3.1 BAR0 Map (LO) 82
4.3.2 BAR0 Map (HI) 83
4.3.3 FPGA Configuration 83
4.3.4 Readback 84
4.4 CLOCK CONTROL 84
4.4.1 Synthesizer Frequencies 84
4.4.2 Clock Sources 85
4.5 COMPACTFLASH INTERFACE 85
4.5.1 Main.txt 86
4.5.2 Unimportant CompactFlash Hardware Notes 89
4.6 USB 89
4.6.1 Configuring an FPGA 89
4.6.2 Readback 90
4.7 CONFIGURING THE "PCI EXPRESS" FPGA 91
4.8 CONFIGURATION REGISTERS 91
4.8.1 Undocumented controls 93
4.9 FIRMWARE 93

5 CLOCK NETWORK 94

5.1 GLOBAL CLOCKS 94
5.1.1 Clock Test points 95
5.2 G0, G1, G2 CLOCKS 96
5.2.1 Synthesizer Circuit 97
5.3 EXT CLOCKS 99
5.3.1 Daughtercard zero-delay mode 99
5.3.2 SMA input 100
5.4 MB CLOCK 101
5.5 FBA AND FBB CLOCKS 101
5.6 PCI EXPRESS REFCLK NETWORK 103
5.7 NON-GLOBAL CLOCKS 103
5.7.1 Clock TP 103
5.7.2 Ethernet Clock 104
5.7.3 DDR2 Clocks 105
5.7.4 SMA Clock B and E 105
5.8 CLOCK USE NOTES 106
5.8.1 Achieving Zero clock-to-out 106
5.8.2 Forwarding Clocks FPGA-to-FPGA 106

6 TEST POINTS 110

7 USB INTERFACE 112

7.1 VENDOR REQUESTS 113
7.1.1 VR_CLEAR_FPGA 113
7.1.2 VR_SETUP_CONFIG 114
7.1.3 VR_END_CONFIG 114
7.1.4 VR_SET_EP6TC (Read buffer size) 114
7.1.5 VR_MEM_MAPPED (Configuration Registers) 114
7.2 MAIN BUS ACCESSES 114
7.2.1 Note about Endpoint Terminology 115
7.2.2 Performance 116
7.3 FPGA CONFIGURATION MODE 116
7.4 MASS STORAGE DEVICE MODE 117
7.5 FIRMWARE UPDATE MODE 117
7.5.1 Activity LED 117
7.6 HARDWARE 117
7.7 TROUBLESHOOTING 117
7.7.1 USB Controller Freezes 117

8 FPGA Q RESOURCES 118

8.1 FPGA A INTERCONNECT 118
8.2 UNUSABLE IO 118
8.3 ROCKETIO (“MGT”, “GTP”, “GTX”) 118

8.4 SPI FLASH 119
8.5 LEDS 119
8.6 RS232 119
8.7 SYNTHESIZER 119

9 PCI EXPRESS INTERFACE 119

9.1 HOST INTERFACE, ELECTRICAL 121
9.1.1 Power 122
9.1.2 PCI-X 122
9.2 HOST INTERFACE, MECHANICAL 122
9.3 PROVIDED "FULL-FUNCTION PCI EXPRESS ENDPOINT" 123
9.3.1 BAR 0 Access 124
9.3.2 BAR 1-5 Access 125
9.3.3 DMA Channels 0 and 1 125
9.3.4 DMA Posted Mode 125
9.3.5 DMA Main Bus 126
9.3.6 Electrical 126
9.3.7 Timing 126
9.3.8 FPGA Interface 127
9.3.9 Host Interface, Software 128
9.4 OTHER PROVIDED DESIGNS FOR THE LXT 131
9.4.1 No design 131
9.4.2 PIPE 132
9.4.3 Slowdown PIPE Core 132
9.5 TROUBLESHOOTING 133

10 UNUSABLE PINS 133

10.1 ADJACENT ROCKETIO 134
10.2 NO CONNECT 134
10.3 CONFIGURATION 134
10.4 VREF/DCI 134

11 SYSTEM MONITOR/ADC 134

12 RESET 135

12.1 POWER RESET 135
12.2 USER RESET 136

13 JTAG 136

13.1 FPGA JTAG 136
13.1.1 Compatible Configuration Devices 137
13.1.2 ChipScope 137
13.2 FIRMWARE UPDATE HEADER 138
13.3 TROUBLESHOOTING 138

14 RS232 INTERFACE 138
14.1.1 Configuration RS232 139
15 TEMPERATURE SENSORS 139
16 ENCRYPTION BATTERY 140
16.1 EXTERNAL BATTERY 141
17 LED INTERFACE 142
17.1 CONFIGURATION SECTION LEDS 142
17.2 USER LEDS 143
17.3 ETHERNET LEDS 145
17.4 POWER LEDS 145
17.5 UNUSED LEDS 146
18 DDR2 DIMM SOCKETS 146
18.1 POWER 147
18.1.1 Interface Voltages 147
18.1.2 Changing the DIMM voltage 148
18.1.3 DIMM warning LED 149
18.2 CLOCKING 150
18.2.1 DQS timing 151
18.2.2 Serial Interface 151
18.2.3 Timing 151
18.3 COMPATIBLE MODULES 152
18.4 INCOMPATIBLE MODULES 152
18.5 TEST POINTS 152
19 FPGA INTERCONNECT. 153
20 MAIN BUS 155
20.1 MB SIGNALS 155
20.1.1 MB vs. MainBus Disambiguation 156
20.1.2 Electrical 156
20.1.3 Timing 156
20.2 ERROR CODES 156
20.3 MAIN BUS FPGA INTERFACE 157
20.3.1 mb_target.v 158
20.3.2 Conventional Memory map 158
21 ETHERNET 159
21.1 RGMII 159
21.1.1 Electrical 160
21.1.2 Timing 160

21.2 CONFIGURATION REGISTERS 161
21.3 MII INTERFACE 162
21.4 EXTERNAL EPROM 162
21.5 EPROM PHY CONFIGURATION 162
21.6 JTAG 163
21.7 ETHERNET MAC 163

22 EPROM 163

23 SPI FLASH 164

23.1 ON FPGAS A AND B 164
23.2 ON FPGA Q 165

24 MICTOR CONNECTORS 165

24.1 FPGA A MICTOR 166
24.2 FPGA B MICTOR 167
24.3 MAINBUS MICTOR 168

25 POWER 169

25.1 POWER 12V 170
25.2 POWER 3.3V 170
25.3 POWER 2.5V 170
25.4 GROUND 170
25.5 VOLTAGE REGULATION 170
25.6 POWER CONNECTIONS 171
25.7 POWER MONITORS 171
25.8 POWER THRU-HOLE ACCESS POINTS 172
25.9 POWER MEASUREMENT TP 173
25.10 HEAT 173
25.10.1 Fans 174
25.10.2 Removing Heatsinks 174
25.10.3 Fan Tachometers 174

26 CONNECTORS 176

26.1.1 Comments 176

27 MECHANICAL 177

28 DAUGHTERCARD HEADERS 178

28.1 DAUGHTER CARD PHYSICAL 179
28.1.1 Daughter Card Locations and Mounting 180
28.1.2 Standard Daughtercard Size 182
28.1.3 Insertion and removal 182
28.2 DAUGHTER CARD ELECTRICAL 183
28.2.1 Pin assignments 190

28.2.2 CC, VREF, DCI *192*
28.2.3 Global clocks *192*
28.2.4 Timing and Clocking *193*
28.2.5 Incorrect Clocking Methods *197*
28.2.6 Power and Reset *199*
28.2.7 VCCO Voltage *199*
28.2.8 VCCO bias generation *200*
28.3 ROLLING YOUR OWN DAUGHTERCARD 200

29 TROUBLESHOOTING 201

29.1 THE BOARD IS DEAD 201
29.2 THE BOARD DOES NOT RESPOND OVER PCI EXPRESS 201
29.3 THE BOARD DOES NOT RESPOND OVER USB 202
29.4 THE FPGAS WON'T PROGRAM 202
29.5 MY DESIGN DOESN'T DO ANYTHING 202
29.6 THE DCMS WON'T LOCK 203
29.7 IT'S SO WEIRD… IT'S LIKE SOMETIMES WHEN I PROGRAM MY FPGAS, THE SIGNALS BETWEEN THE FPGAS ARE DELAYED BY ONE CLOCK CYCLE. THEN, WHEN I HIT THE RESET BUTTON, SOMETIMES IT STARTS WORKING AGAIN. 203
29.8 MY PACEMAKER STOPS WORKING WHEN I INCREASE THE CLOCK FREQUENCY 203
29.9 THE SIGNAL ON MY BOARD IS GOING BAT CRAZY ON MY OSCILLOSCOPE 203

CHAPTER 5: REFERENCE DESIGN 205

1 PURPOSE 205

1.1 INTERFACES USED BY REFERENCE DESIGN 205
1.2 INTERFACES NOT USED BY THE REFERENCE DESIGN 206

2 HARDWARE TESTS 206

2.1.1 Testing PCI Express interface *206*
2.1.2 Testing FPGA-to-FPGA interconnect *206*
2.1.3 Testing DDR2 Interfaces *206*
2.1.4 Testing USB *207*
2.1.5 Testing Ethernet *207*
2.1.6 Testing Daughtercard Connectors *207*

3 REFERENCE DESIGN TYPES 207

3.1 MAIN TEST 207
3.2 LVDS 208
3.3 SINGLE FAST 208
3.4 V5 INTERCONNECT 208
3.5 ETHERNET 208
3.6 HEADER 208

4 USING THE REFERENCE DESIGN 208

4.1 REFERENCE DESIGN MEMORY MAP 208

5 INTERCONNECT (SINGLE) 210

5.1 USING THE DESIGN 210
5.2 RUNNING THE TEST 211
5.3 DDR2 INTERFACE 211
5.4 PROVIDED FILES 211
5.5 USING THE DESIGN 211
5.6 RUNNING THE TEST 212
5.7 CLOCK COUNTERS 212
5.8 LEDS 212
5.9 SIMULATING THE REFERENCE DESIGN 212

6 LVDS REFERENCE DESIGN 213

6.1 PROVIDED FILES 213
6.2 USING THE DESIGN 213
6.3 RUNNING THE TEST 213
6.4 IMPLEMENTATION DETAILS 214
6.4.1 Lane Alignment 214
6.4.2 Funny Banks 214

7 PCIE INTERFACE REFERENCE DESIGN 215

7.1 PROVIDED FILES 215
7.2 USING THE DESIGN 215
7.3 RUNNING THE TEST 215

1 COMPILING THE REFERENCE DESIGN 216

1.1 THE XILINX EMBEDDED DEVELOPMENT KIT (EDK) 216
1.2 XILINX ISE 216
1.3 THE BUILD UTILITY: MAKE.BAT 217
1.4 BITGEN OPTIONS 217
1.5 VHDL 218

CHAPTER 6: ORDERING INFORMATION 219

1 HOW TO ORDER 219

2 OPTIONAL EQUIPMENT 219

2.1 COMPATIBLE DINI GROUP PRODUCTS 219
2.1.1 Interface Boards 219
2.1.2 Memories 219
2.1.3 Daughter cards 220
2.2 COMPATIBLE THIRD-PARTY SOFTWARE 221
2.3 COMPATIBLE THIRD-PARTY HARDWARE 221

3 COMPLIANCE DATA 222

3.1 DISCLAIMER 222
3.2 COMPLIANCE 222
3.2.1 FCC EMI 222
3.2.2 PCIe-SIG 223
3.3 ENVIRONMENTAL 223
3.3.1 Temperature 223
3.4 EXPORT CONTROL 223
3.4.1 Lead-Free 223
3.4.2 The USA Schedule B number based on the HTS 223
3.4.3 Export control classification number ECCN 224

2 List of Figures

Figure 1 - DN9200K10PCIE8T – Heat sinks negligently left uninstalled. 19
Figure 2 – An example circuit on the board. 27
Figure 3 – How that circuit appears on the customer netlist. 27
Figure 4 - An engineer demonstrates use of a grounding wrist strap ..... **Error! Bookmark not defined.**
Figure 5 - DN9200K10PCIE8T stuff you need to know about to get started 33
Figure 6 - A six-pin PCI Express "Graphics Power" adapter 36
Figure 7 - A power supply "starter" 36
Figure 8 – Figure 8 ..... **Error! Bookmark not defined.**
Figure 9 - RS232 Output 38
Figure 10 - LEDs 39
Figure 11 - Driver installation Wizard 39
Figure 12 - USB Controller Window 40
Figure 13 - USB Controller Log Output 41
Figure 14 - USB Controller Log Output 43
Figure 15 - Splash screen 44
Figure 16 - AETest Main Menu 45
Figure 17 - Memory Menu 45
Figure 18 - JTAG Headers 46
Figure 19 - iMPACT connected to FPGA JTAG 47
Figure 20 - USB Controller Main Window 51
Figure 21 - Refresh Button ..... **Error! Bookmark not defined.**
Figure 22 - Enable USB Button ..... **Error! Bookmark not defined.**
Figure 23 - USB Controller complains if board is not detected 52
Figure 24 - Configuring FPGAs 53
Figure 25 - AETest splash screen 60
Figure 26 - AETest main menu 61
Figure 27 - AETest PCI menu 61
Figure 28 - AETest memory menu ..... **Error! Bookmark not defined.**
Figure 29 - AETest Testing menu ..... **Error! Bookmark not defined.**
Figure 30 - Firmware Update Header 66
Figure 31 - iMPACT Window 67
Figure 32 - Switch S2 69
Figure 33 - USB Controller Firmware Update Mode 69

Figure 34 - JTAG Headers 70
Figure 35 - DN9200K10PCIE8T Block Diagram 73
Figure 36 - DN9200K10PCIE8T LX110 Block Diagram 75
Figure 37 - LX Selection Guide 76
Figure 38 - LXT FXT Selection Guide 76
Figure 39 - Config Section Block Diagram 79
Figure 40 - Serial Port Headers 80
Figure 41 - DONE LED circuit 81
Figure 42 - EXT0 EXT1 Circuit 85
Figure 43 - CompactFlash card socket 86
Figure 44 - Main.txt Commands 88
Figure 45 - Spartan "Firmware" JTAG Chain 93
Figure 46 - Clock network block diagram 95
Figure 47 - Clock Test points 95
Figure 48 - Clock G network synthesizer circuit 97
Figure 49 - EXT clock sources diagram 100
Figure 50 - EXT0 SMA locator 101
Figure 51 - EXT0 SMA circuit 101
Figure 52 - FBA typical use 102
Figure 53 - FBA typical use with synchronization 102
Figure 54 - Clock Testpoint circuit 104
Figure 55 - Clock Test point locator 104
Figure 56 - SMA circuit 105
Figure 57 - SMA locator 106
Figure 58 - Not using GCLK pins 107
Figure 59 - Not using an external feedback 108
Figure 60 - Two divide DCMs 108
Figure 61 - Outputting a clock with an assign statement 109
Figure 62 - Cascading DCMs 109
Figure 63 - DCM on same reset as logic 110
Figure 64 - USB locator 112
Figure 65 - PCI SIG Compliance Base Board 118
Figure 66 - FPGA Q LEDs 119
Figure 67 - PCI Express block diagram 120
Figure 68 - PCI Express circuit 121
Figure 69 - PCI Express eye diagram 122
Figure 70 - Full function design block diagram 123
Figure 71 - FPGA A to Q clocking diagram 127
Figure 72 - PIPE design block diagram 132
Figure 73 - PIPE Slowdown block diagram 133
Figure 74 - Sysytem monitor circuit 134
Figure 75 - FPGA JTAG circuit 136
Figure 76 - FPGA JTAG locator 137
Figure 77 - FPGA JTAG block diagram 137
Figure 78 - RS232 circuit 138
Figure 79 - RS232 locator 139
Figure 80 - Battery locator 141
Figure 81 - battery circuit 142
Figure 82 - LED circuit 143
Figure 83 - LED locator 144
Figure 84 - Ethernet locator 145

Figure 85 - Power fail LED locator 145
Figure 86 - Unused LED locator 146
Figure 87 - DIMM block diagram 147
Figure 88 - DIMM Voltage selection circuit 148
Figure 89 - DIMM Voltage locator 149
Figure 90 - DIMM warning LED locator 149
Figure 91 - DIMM clock diagram 150
Figure 92 - DIMM signal test point locator ..... **Error! Bookmark not defined.**
Figure 93 - Interconnect block diagram 153
Figure 94 - Main Bus block diagram 155
Figure 95 - Inaccurate Mani Bus read timing 157
Figure 96 - Inaccurate Main Bus write timing 158
Figure 97 - Ethernet locator 159
Figure 98 - Ethernet timing 160
Figure 99 - 1000Base-T circuit 162
Figure 100 - EPROM circuit 164
Figure 101 - SPI Flash circuit 164
Figure 102 - SPI Flash circuit Q 165
Figure 103 - Mictor locator 166
Figure 104 - Mictor cable 166
Figure 105 - Mictor A circuit 167
Figure 106 - Mictor B circuit 167
Figure 107 - MainBus mictor locator 168
Figure 108 - Main Bus mictor circuit 168
Figure 109 - Board power topology diagram 169
Figure 110 - PCI Express graphics power locator 171
Figure 111 - Power Test points 172
Figure 112 - Power Fail LED locator 173
Figure 113 - Power probe point circuit 173
Figure 114 - Heatsink fan locator 174
Figure 115 - Fan tachometer circuit 175
Figure 116 - Fan power locator 175
Figure 117 - Mechanical drawing 177
Figure 118 - Ground rail locator 178
Figure 119 - Daughter card locator 178
Figure 120 - Daughter card block diagram 179
Figure 121 - Mechanical Drawing 180
Figure 122 - Daughter card side mechanical 181
Figure 123 - DNMEG_EXT mechanical 181
Figure 124 - Standard daughter card dimensions 182
Figure 125 - Daughter card installation step 1 183
Figure 126 - Install Daughter card step 2 183
Figure 127 - Daughter card pinout diagram 191
Figure 128 - Daughter card clock pin functions 193
Figure 129 - Daughtercard clocking local 194
Figure 130 - Daughter card clocking global 195
Figure 131 - Daughter card clocking source synchronous 196
Figure 132 - Daughter card clocking skew tolerant 197
Figure 133 - Daughter card Clock forwarding fail 198
Figure 134 - Daughter card clocking PLL cascade fail 198
Figure 135 - Tacoma Narrows Fail ..... **Error! Bookmark not defined.**

Figure 136 - MEG Array power circuit 199
Figure 137 - MEG Array bias circuit 200
Figure 138 - Dini Group corporate strategy diagram 205
Figure 139 - LVDS Reference design clocking global 214
Figure 140 - LVDS Reference design clocking local 215
Figure 141 - Disclaimer block diagram 222

Chapter 1: Introduction

Congratulations on your purchase of the DN9200K10PCIE8T logic emulation board. If you are unfamiliar with Dini Group products, you should read Chapter 2, Quick Start Guide to familiarize yourself with the user interfaces the DN9200K10PCIE8T provides.

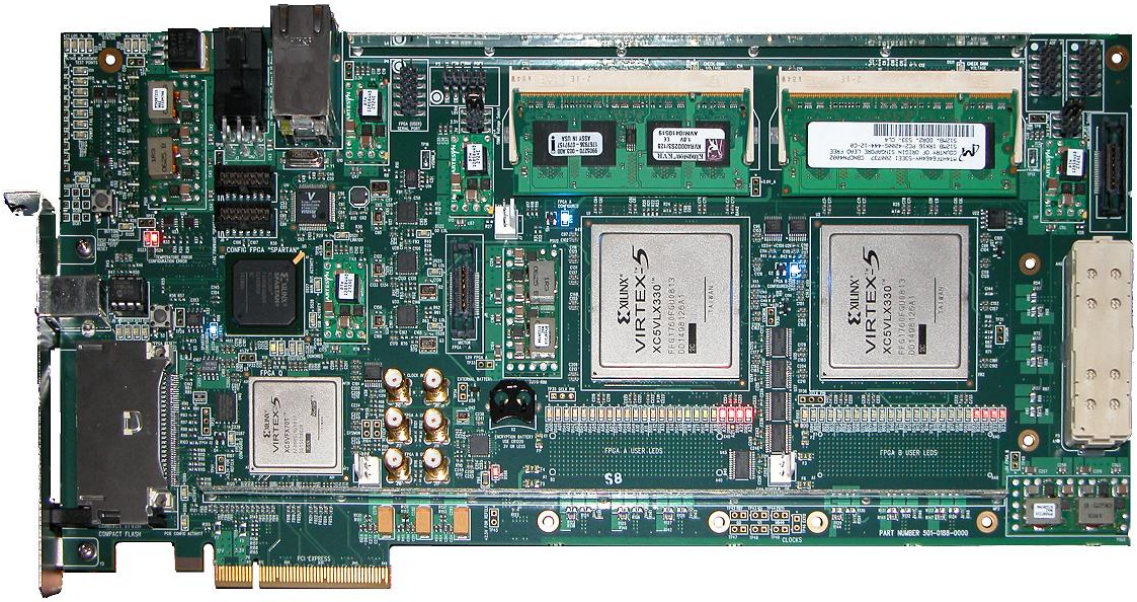


Figure 1 DN9200K10PCIE8T – Heat sinks negligently left uninstalled.

1 Manual Contents

This manual contains the following chapters:

1.1 Introduction

Reader's Guide to this manual; List of available documentation and resources; Section 1 contains a list of the manual contents, including the introduction.

1.2 Quick Start Guide

This chapter includes step-by-step instructions for powering on the DN9200K10PCIE8T for the first time. It will guide you through using the board's most important features. For users very familiar with FPGA boards, this is likely the only part of the manual that will need to be read completely. The rest of the book can be used for reference.

1.3 Controller Software

A summary of the functionality of the provided software; Implementation details for the remote USB board control functions and instructions for developing your own USB host software

1.4 Hardware

This chapter is to be used as a reference for use of the individual circuits available to the user. When implementing an interface on the FPGA, you should read its corresponding section in this chapter in conjunction with the parts datasheets and the board schematic.

1.5 The Reference Design

This chapter will describe parts of the provided FPGA code and project files that seem like they are important. Users very familiar with FPGA boards probably will not use the reference designs. People new to FPGA board development might want to start from one of the example designs.

1.6 Ordering Information

This chapter contains a list of the available options and available optional equipment; some suggested parts and equipment available from third party vendors; Also information about the board that has nothing to do with actually using the board.

2 Audience

Certain assumptions are made about the audience of this manual. Below is a list of the prerequisite skills to successfully use the board and the manual. A resource is suggested for further reading is necessary.

The reader is fluent in Verilog or VHDL.
A Verilog HDL Primer by Jayaram Bhasker
www.amazon.com

The reader understands how to calculate required timing parameters on an electrical interface using an IC manufacturer's part datasheet.

The reader knows how to implement an HDL design using the Xilinx XST design flow.
http://www.xilinx.com/support/software_manuals.htm

3 Conventions

This document uses the following conventions. An example illustrates each convention.

3.1 Notations

Prefix "0x" The radix on numbers is usually decimal. By convention, I've started radix-16 numbers with "0x"

Postfix "#" and "n" and "m" On signal names or logical values whose names end in # or N usually have an inverted logical value. Or, in the case of physical signals on the board, have an active state represented by a low voltage.

3.2 File paths

Paths to documents included on the User CD are prefixed with "D:\". This refers to your CD drive's root directory when the User CD is inserted in your Windows computer.

For some things to work correctly (compilations, executables, projects) you will probably need to copy the entire contents of the User CD to your hard drive. In this case, D:\ will to refer to the path of the copy on your hard drive. Due to limitations of the Xilinx ISE software in Windows, we recommend a path without space characters in it. (Bad places include C:/Documents and Settings/username/Desktop/)

3.3 Physical Dimensions

By convention, the board is oriented as shown in the above board photo, with the "top" of the board being the edge with the Ethernet RJ45 connectors. The "right" edge is near FPGA C and F. The "left" side is the side with the PCIe bracket. "Top" side refers to the side of the PWB with FPGAs and fans; the "back" side is the side with the three daughtercard connectors. The reference origin of the board is the center of the lower PCI bracket mounting hole.

All physical dimensions are given in millimeters, when no units are specified.

3.4 Part Pin Names

References to individual part's pin are given in the form <*X*><*Y*>.<*Z*>; The <*X*> is one of: U for ICs, R for resistors, C for capacitors, P or J for connectors, FB or L for inductors, TP for test points, MH for mounting structures, FD for fiducials, BT for sockets, DS for displays (light-emitting diodes), F for fuses, PSU for power supply modules, Q for discrete semiconductors, RN for resistor networks, G for oscillators, X for sockets, Y for crystals and the PCIe bezel. <*Y*> is a number uniquely identifying each part from other parts of the same class. <*Z*> is the pin or terminal number or name, as defined in the datasheet of the part. Datasheets for all standard and optional parts used on the DN9200K10PCIE8T are included in the Document library on the user CD.

3.5 Schematic Clippings

Partial schematic drawings are included in this document to aid quick understanding of the features of the DN9200K10PCIE8T. These clippings **have been modified** for clarity and brevity, and **may be missing** signals, parts, net names, labels and connections. Unmodified Schematics are included in the User CD as a PDF.

Designing interface logic for external parts on this board will certainly require at least some use of the schematic. Use the PDF search feature to search for nets and parts.

4 Glossary

In this manual, references are made to these things that may have no meaning to you:

Spartan……. Spartan refers to the Spartan-3 FPGA device used by the DN9200K10PCIE8T to perform configuration circuit functions. It is used also interchangeably with "configuration circuit". This FPGA is not intended to be used by you.
Config FPGA
U0

FPGA Q……. There are four FPGAs on this board: FPGA A, FPGA B, FPGA Q and the Spartan. The first three are intended for the user to use. The Spartan is reserved for board control and should not be considered for emulating your logic. Dini Group provides bit files that can be used in FPGA Q in bitstream from (we do not provide the RTL for some of these). These bit files implement PCI express and can be used as a ready-to-go PCI Express endpoint, or you may chose to use FPGA Q as a third user FPGA. If you need PCI Express, in this case, you will have to implement your own PCI Express endpoint or uses the Xilinx Block+ core.
V5T
PCI Express FPGA
LXT
FXT
U3
"QL"

ISE……. These are software products provided by Xilinx
bitgen
iMPACT
XST
EDK
CoreGen
MIG

Bitfile……. This is the contents of the SRAM that controls the FPGA's internal behavior. The data file that contains this data is a .bit file and is generated by Xilinx bitgen
Configuration Stream
.bit file

DCM……. These terms refer to features of the Virtex-5 FPGA that it assumed that you know about. Understanding the function (and using) all of these primitives is definitely required to make your design work properly.
DCI
BUFG
DIFF_TERM
ODDR
IOB

MGT…….. These all refer to features of the Virtex-5 FPGA that is assumed that you know about. Understanding the function (and using) these features may be required to make your design work properly. See the Virtex-5 user guide.
GTP
GTX
BUFR
BUFIO
OSERDES
IDELAY

LVDS…….. These refer to signaling standards (voltage levels) that are required to make some interfaces external to the FPGA work properly. When you know the IO standard of external signal that must be driven, it is usually sufficient to simply select the corresponding output and input standard in the FPGA. In the case when this is not possible, you are expected to look up the drive standard and ensure that the selected FPGA output class is appropriate.
SSTL
LVCMOS
LVDCI

UCF…….. This is the something constraint file. This along with your RTL specifies the behavior of the FPGA, once it's configured. The UCF contains information about the IO pins electrical and timing behavior. Using a UCF is required. Your design will not work without one.
LOC
IOSTANDARD
DRIVE

Net……. These names all refer to a physical conductor on the circuit board connecting pads of ICs on the board.
Signal
Plane
Rail
Transmission Line

GND…….. GND is a net on the DN9200K10PCIE8T. All absolute voltages given are offsets with respect to this net. It may also refer to a signal or net whose measured voltage is equal to this net.
ground
grounded
0V

AppNote……. These are publications from Xilinx that are available on the Xilinx website.
XAPP

Verilog…….. This is the code that you put in an FPGA
VHDL
RTL
Core
IP
Design

PCIE…….. PCI Express
Gen2……... PCI Express specification revision 2.0
Mux……Multiplexer

MBs……... Mega Byte per second. (1,000,000 bytes)
MB……. Mega Bytes (1,048,576 bytes)
GBs…… Giga Byte per second (1,000,000,000 bytes)
Mbs……Mega bit per second (1,0000,000 bits)
Gbs…….Giga bit per second (1,000,000,000 bits)
MTs……... Mega Transfers per second. Same as MHz, except it is not ambiguous with respect to spectral power content like MHz.
MHz…….. Megahertz; "One million cycles per second" (1,000,000). Can either to the number of transactions per second, or the spectral content of the synchronizing clock of a signal, which is half the transfer rate.
DDR…….. "Double data rate". This probably refers to a specific memory interface specification for DRAMs. It can also refer to the practice of running the clock on a synchronous system at half-frequency to improve the signal integrity of the clock.

5 Resources

The following electronic resources will help you during development with your board.

5.1 User CD

The User CD contains all the electronic documents required for you to operate the DN9200K10PCIE8T. These include schematics, the user manual, FPGA reference designs, and datasheets. The directory structure of the CD is as follows

Config_Section_Code\	The DN9200K10PCIE8T firmware source code. This code is provided in case Dini Group gets hit by a meteor. Under other circumstances, you shouldn't need to look in this directory.
Datasheets\	A datasheet for every part used on the board. You will need these to interface successfully with resources on the DN9200K10PCIE8T
DNMEG_xxx\	Information about some common (optional) daughtercards
DNPCIE_CBL_CableAdapterDaughtercard\	Information about some common (option) daughtercards
Documentation\Manual\	Contains this document.
Documentation\MEG400_connectio…	Contains a spreadsheet the lists the pinout of all off-the-shelf Dini Group daughter cards.

Documentation\Dini_USB_Spec	Contains information about implementing USB software that interfaces with the board. This document is more detailed about the actual software required in a Windows or Linux application.
FPGA_Reference_Designs\ common\ DN9200K10PCIE8T\ Programming_Files\ opencore\ pcie\	Contains the source and compiled programming files for the Dini group's DN9200K10-PCIe reference design; Also, board description files and simulation models
FPGA_Reference_Designs\common\	Contains code that is used by many Dini Group products. Some subdirectories may not be applicable. This directory must be in the include path of your Xilinx project when compiling the reference design or it won't work very well.
FPGA_Reference_Designs\DN9200K10PCIE8T \	Contains code specific to DN9200K10PCIE8T; Also contains partitioning models for some automatic partitioning tools.
FPGA_Reference_Designs\pcie	Contains information and code for interfacing with the provided PCI Express endpoint bitstream for FPGA Q.
PCI_Software_Applications\AETest\	Source and binaries for the provided PCI Express host software.
Schematics\Rev_01\	Contains a PDF version of the board schematic. Search the PDF using control-F. Also contains an ASCII netlist of the board.
USB_Software_Applications\ driver\ AETEST_USB\ USBController\	Contains source and binaries for the provided USB-hosted controller applications.

5.2 Dinigroup.com

The most recent versions of the following documents are found on the product web page http://dinigroup.com/DN9200k10PCIe-8T.php

User's Manual (this document)
Board Errata (if exists)
Wild marketing promises
Updates to the constantly "improving" USB Controller Windows executable
Links to other things you might buy

5.3 Errata List

The Errata sheet (available at www.dinigroup.com) lists all cases where the DN9200K10PCIE8T is found to have failed to meet advertised specifications, or where an error in schematics or documentation is likely to cause a difficult-to-debug error by the user.

5.3.1 Existing Errata

The errata list was empty at August 1, 2008

5.4 Reference Design

The reference design implements something on every user IO in the device. For many users, the UCF provided with the reference design is the primary reference document.

5.5 Schematics and Netlist

Unmodified Schematics are included in the User CD as a PDF. Use the PDF search feature to search for nets and parts.

5.5.1 Netlist

In lieu of providing a machine-readable version of the schematic, the Dini Group provides a text netlist of the board. This netlist contains all nets on the board that connect to user IO on any FPGA. It does not contain all nets on the board. The schematic is the only provided resource that completely describes the board.

When interfacing with any device or connector on the DN9200K10PCIE8T you should use either the provided .ucf, or the netlist to generate the pinout. The netlist is located on the user CD at D:\Schematics\Rev_01\DN9200K10PCIE8T_customer_netlist.txt

It is in a difficult-to-use "wirelist" format, which is fixed-column-width format. You will probably need to mangle it in Excel to make any use of it.

Remember that logical signals may be represented by multiple nets on the board, for example, a clock signal that has a DC blocking capacitor on it, may only appear in the netlist as a connection to some useless dangling capacitors… but they aren't.

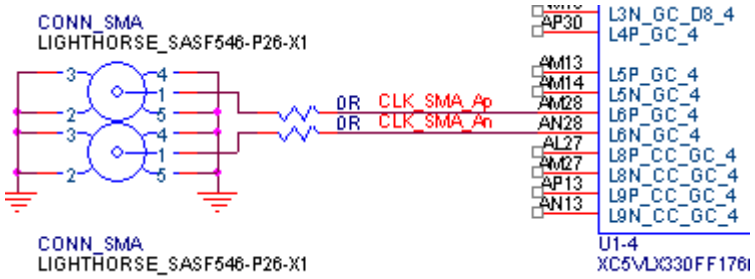


Figure 2 – An circuit on the board.

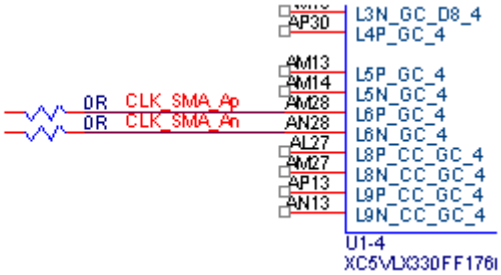


Figure 3 – How that circuit appears on the customer netlist.

5.5.2 Net name conventions

All "power" nets begin with a +, - symbol, or GND

All clock signals begin with "CLK"

Two sides of a differential signal differ by one character "p" or "n". This character is near the end of the net name.

Active low signals end in # or N. In the provided UCF files, the # is replaced by an "N".

5.6 Datasheet Library

Datasheets for all parts used, or interfaced to, on the DN9200K10PCIE8T are provided on the user CD. In order to successfully use the DN9200K10PCIE8T, you will have to reference these datasheets. The interface descriptions given in this user manual typically end with electrical connectivity.

Especially read the Virtex-5 user guide. The copy provided on the user CD is only recent as of the DN9200K10PCIE8T product announcement.

5.7 Xilinx

The internal behavior of the Virtex-5 device is beyond the scope of technical support for this board, although we might happen to know the answer to your questions. Technical questions about the internal operation of the FPGA and ISE software behavior should be directed to a Xilinx FAE. Also use:

WebCase	http://www.xilinx.com/support/clearexpress/websupport.htm
AnswerBrowser	http://www.xilinx.com/xlnx/xil_ans_browser.jsp
ISE Manual	http://www.xilinx.com/support/sw_manuals/xilinx82/index.htm
Virtex 5 Manual(s)	http://www.xilinx.com/support/documentation/virtex-5.htm (Also on the User CD)

5.8 Dini Group Reference Designs

The source code to the reference designs are on the User CD. Please copy and use any code you would like without restriction. The reference designs themselves are intended as examples, and are likely not suitable for a particular purpose. Therefore, support for these products is limited to their ability to demonstrate how certain interfaces might be implemented.

5.9 Board Models

Auspy board partitioning models, other partitioning models, and simulation models for the DN9200K10PCIE8T are provided on the user CD.

D:\FPGA_Reference_Designs\DN9200K10PCIE8T\source\

5.9.1 Base System Builder

There is not a provided BSB file for the board; however creating new projects is not very difficult.

5.9.2 Using Partitioning and 3rd party synthesis tools.

We cannot support directly third party synthesis tools and partitioning tools that we do not have. Therefore, support for these tools must be obtained from the software vendor.

5.10 PCI Express Details

A separate file contains details about the behavior of the LXT "PCI Express FPGA" when it is loaded with our provided "Full function PCI Express endpoint now with DMA™" bitfiles. That document can be found on the user CD here:
D:\FPGA_Reference_Designs\common\PCIE_x8_Interface

5.11 Email and Phone Support

Our phone number is (USA) 858-454-3419.

Dave Palmer	x30	Questions about board hardware, complaints about the user manual
Ivan Yulaev	x12	All other technical questions, complaints about life
Mike Dini	x11	Sales Questions, complaints about employees

Dini Group technical support for products can be reached via email at support@dinigroup.com. If you just want to buy accessories, email sales@dinigroup.com

Please do not send .exe files, .vb files, .zip files containing other .zip files, or certain types of image files as attachments, as we will not receive these emails due to virus scanner ultra technology. Please include the board's serial number in your email. This will allow us to reference our records regarding your board.

Before contacting support for hardware failures, you should complete the following:

1) Follow the debugging steps in the troubleshooting sections at the end of the hardware chapter, and in any applicable interface sections.

2) Test the applicable interface(s) using the provided software and .bit files, to help rule out hardware failures.

Chapter 2: Quick Start Guide

The Dini Group DN9200K10PCIE8T can be used and controlled using many interfaces. In order to learn the use of the most fundamental interfaces of the board (FPGA Configuration, USB data movement, etc.) please follow the instructions in this quick start guide. The guide will also show you how to run the board's hardware test to verify board functionality. (The board has already been tested at the factory).

1 Provided Materials

Examine the contents of your DN9200K10PCIE8T kit. Print this page and check off the following:

☐ **DN9200K10PCIE8T** board

☐ **Compact Flash card** containing the FPGA configuration ".bit" files required to run the hardware test.

☐ **Card reader** USB to Compact Flash

☐ **Adapter Cable** for RS232 (10-pin header to female DB9)

☐ **Adapter cable** for PCI Express "graphics power" connector

☐ **PSU Starter**

☐ **USB cable**; black or zebra-striped

☐ **Mounting hardware** (for daughter cards)

☐ **CD ROM** containing:
- Virtex 5 Reference Designs
- User manual PDF
- Board Schematic PDF
- USB program (usbcontroller.exe)
- PCIe program (Aetest.exe)
- Source code for USB program, PCIe program and DN9200K10PCIE8T firmware
- Board netlist

☐ **Gray Foam**

1.1 System Requirements

Virtex-5 requires ISE 8.2, however this guide is written assuming ISE 10.2.04 is installed. Versions before this may have different steps required which aren't given here. (Just download 10.2)

The board is provided with software that can be used in various versions of Windows or Linux, however in this guide, it is assumed that you have access to an Intel-compatible 32-bit computer with Windows XP SP2 or SP3 installed, USB 2.0 and a PCI Express x16 slot. Otherwise, different steps may be required which aren't given here. (Just borrow the office manager's Windows machine)

It is assumed that you have a Xilinx Platform USB or Platform USB II cable for use with JTAG. Use of this board is possible without this cable; however this guide assumes that you have one. Steps for using JTAG or updating firmware may be different if you do not have this cable. (Just order a Xilinx JTAG cable)

Your life will also be easier with an oscilloscope and a multi-meter.

2 Warnings

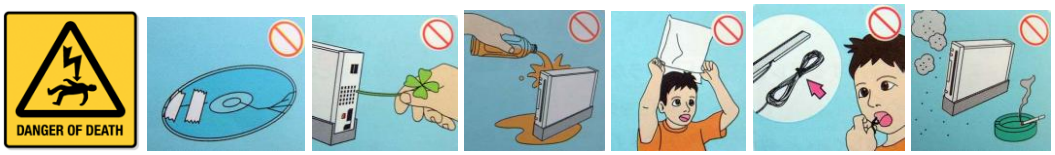
2.1 ESD

The DN9200K10PCIE8T is sensitive to static electricity, so treat the PCB accordingly. The target markets for this product include engineering departments who are familiar with FPGAs and circuit boards. If you are unfamiliar with electrostatic discharge, please go read about it on Wikipedia before touching the board. There are exposed ESD-sensitive points all over the DN9200K10PCIE8T. Shocking one of the exposed IOs of one of the FPGAs could lead to a costly repair or having to pretend like it was like that when you got it. However, if needed, the following web page has an excellent tutorial on the "Fundamentals of ESD" for those of you who are new to ESD sensitive products: http://www.esda.org/basics/part1.cfm

There are two large grounded metal rails on the DN9200K10PCIE8T. The user should grip the board using these rails like a Mawashi.

The 400-pin connectors are not 5V tolerant. In fact, very few exposed surfaces on the board are tolerant of voltages greater than 4V. According to the Virtex 5 datasheets, the maximum applied voltage to any IO signals on the FPGA is the "VCCO" voltage associated with the daughter card. This means you should not try to over-drive IOs in an FPGA interface above the interface voltage specified in this manual.

2.2 Other



Some parts of the board are physically fragile. Take extra care when handling the board to avoid touching the daughtercard connectors. Leave the covers on the daughtercard connectors whenever they are not in use. Use mounting hardware to secure daughtercards.

Surface mount headers with cables attached to them will eventually damage the board when your chair rolls over the cable. If you have cables attached to your board, use cable ties.

3 Pre-Power on Instructions

Most of the cables and connectors on the board are not suitable for hot-swap and should therefore be connected before the board powers on.

The image below represents your DN9200K10PCIE8T. You will need to know the location of the following parts referenced in this chapter.

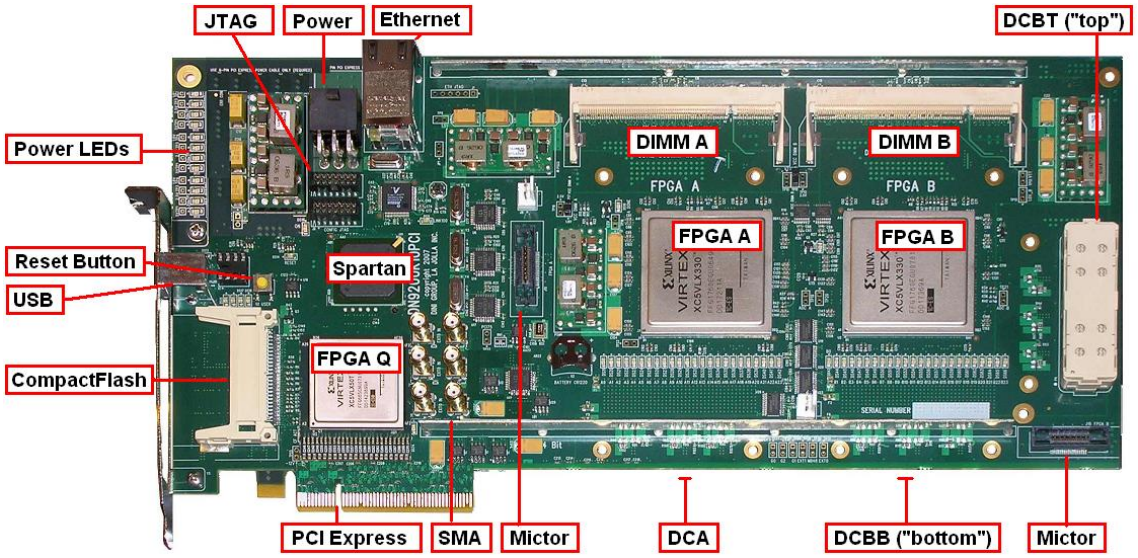


Figure 4 DN9200K10PCIE8T stuff you need to know about to get started.

The FPGAs on the board are named "FPGA A", "FPGA B", FPGA C, FPGA D, FPGA E, and FPGA F as shown in the above photo. The "FPGA Q" is Virtex 5 LX50T.

To begin working with the DN9200K10PCIE8T, follow the steps below.

3.1 Install Memory (optional)

The DN9200K10PCIE8T comes packaged without memory installed. The board does not need memory to run, however the hardware test might report failure on the DDR2 sockets if you do not install some now.

The reference design supports DDR2 SODIMM modules in any densities up to 4 GB (more than 4 GB is not tested). If you find an incompatible DIMM, email us the part number so we can add support for it.

Install the memory in sockets DIMMA and DIMMB

3.2 Prepare configuration files

The DN9200K10PCIE8T can read FPGA configuration data from a CompactFlash card. To program the FPGAs on the DN9200K10PCIE8T, you can place FPGA design files (with a .bit file extension) on the root directory of the CompactFlash card file using the provided USB card reader.

The DN9200K10PCIE8T ships with a 256MB Compact Flash card preloaded with the Dini Group reference design. These "bit" files can also be found on the User CD. You can also compile the reference design source (provided on the CD) and place the generated .bit files on the Compact Flash card.

Insert the provided Compact Flash card labeled "Reference Design" into your USB card reader. Make sure the card contains at least these three files:

FPGA_A.bit (if FPGA A stuffed)
FPGA_B.bit (if FPGA B stuffed)
main.txt

The files FPGA_A-B.bit are files created by the Xilinx program bitgen, part of the ISE 9.2 tools. The file main.txt contains instructions for the DN9200K10PCIE8T configuration controller, including which FPGAs to configure, and to which frequency the global clock networks should be automatically adjusted.

An example main.txt file can be found on the provided CompactFlash card, or on the user CD.

3.3 Insert the Compact Flash card

This step involves inserting the CompactFlash card into the DN9200K10PCIE8T's CompactFlash slot. No further advice is given.

3.4 Install DN9200K10PCIE8T in computer (optional)

If you plan to use the DN9200K10PCIE8T in a PCI express slot, install it now. Do this with power turned off. I do not think this is hot-swappable.

If you are not using the DN9200K10PCIE8T in a PCIe Express slot, skip this step. The board may instead be operated table-top. The DN9200K10PCIE8T is compatible with PCIe-Express 1, 4,8, or 16-lane slots. To physically fit the board into a 4x or 1x slot will require an adapter card, such as those available from Catalyst.

If you skip this step, then AETest cannot be used.

3.5 Connect RS232 Cable

The configuration controller displays status messages to an RS232 terminal. If (when) something goes wrong with configuration, this terminal will output error messages. Normally, you would only connect this cable when something is not working and you want to debug the problem.

Use the provided ribbon cable to connect the MCU RS232 port (P3) to a computer serial port to view feedback from the configuration circuitry during FPGA configuration. Run a serial terminal program on your PC (On Windows you can use HyperTerminal Start->Programs->Accessories->Communications->HyperTerminal) and make sure the computer serial port is configured with the following options:

Bits per second: 19200
Data bits: 8
Parity: None
Stop Bits: 1
Flow control: None
Terminal Emulation: VT100 (or none, if available)

HyperTerminal is a poor program. You can use putty or SecureCRT from Vandyke software if you are a less tolerant person.

3.6 Connect USB Cable

Use the provided USB cable to connect the DN9200K10PCIE8T to a Windows computer (Windows XP or Vista is recommended).

If your board is installed in a PCIe slot, you can connect USB from the same host computer if you wish. A different computer is also okay.

3.7 Connect Power cable

The power cable connected to J3 is required. If you do not plug a cable in here, the board will not power on. This is true whether or not the board is installed into a PCI Express slot. Most new computer power supplies have a 6-pin "PCI Express Graphics" power connector. If yours does not, you can use the provided adapter cable.

Figure 5 - A six-pin PCI Express "Graphics Power" adapter

If you are operating desk-top, and not in a motherboard, then you will need a standalone computer power supply (not provided). Your power supply might not turn on if its 20 or 24-pin "motherboard" power connector is not connected to anything. In this case, connect the provided PSU starter to the PSU.

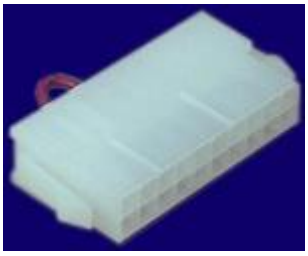


Figure 6 - A power supply "starter"

3.8 Daughter Cards

I know you want to plug your daughter cards in right now, but let's wait until you are familiar with the board first. Also note that these daughtercard interfaces were specifically designed for very high speed, which means they are also specifically designed to break easily. Read the "Hardware" chapter about how to properly install daughter cards before trying it.

4 Power on Instructions

Turn on the Desktop computer power supply (for desktop operation) or the computer (PCIe operation).

When the DN9200K10PCIE8T powers on, it automatically loads Xilinx FPGA design files (ending with a .bit extension), found on the CompactFlash card in the CompactFlash slot into the FPGAs, according to the instruction in the main.txt file on the CompactFlash card.

This process may take 5 or 10 seconds. As each FPGA is configured a nearby blue "DONE" LED will light.

4.1 View configuration feedback over RS232

The purpose of the "MCU" RS232 port is to allow you to determine why the board is not behaving how you expect. There are a few controls available over RS232; however most people do not use them.

As the DN9200K10PCIE8T powers on, your RS232 terminal (connected to P3) will display information about the Configuration process. If FPGAs ever fail to configure using the Compact Flash card, this is the best place to look for help.

A typical RS232 power-on session is given below.

DINI GROUP FLP EEPROM VERSION NEW	Running out of EPROM.
No USB Cable detected	
Rebooting from flash. Please wait.	
DN9200K10PCIE8T FLASH BOOT	Running out of Flash
G0 CHECK: PASS	Hardware checks.
G1 CHECK:PASS	
G2 CHECK:PASS	
……	Hardware self-test
……..	
FPGAs Found	
A B Q	
Resetting CompactFlash: DONE	
Configuration Files on card:	Displays which files were found on the CompactFlash card.
FPGA A: FPGA_A.BIT	
FPGA B: FPGA_B.BIT	
OPTIONS:	
Message Level:2	
SanityCheck: ON	
**********CONFIGURING FPGA A**********	Configuring FPGA A according to main.txt
Sanity Check:pass	
Bit File Properties	
Name: FPGA_A.BIT	
File Size: 009806AB bytes	
Part: 5vlx330ff1760	
Date: 2007/12/20	
PASS	
…….	
……..	
DONE CONFIGURING A	
**********CONFIGURING FPGA B**********	Configuring FPGA B according to main.txt
Sanity Check:pass	
Bit File Properties	
Name: FPGA_B.BIT	
File Size: 009806AB bytes	
Part: 5vlx330ff1760	
Date: 2007/12/20	
PASS	
…….	
……..	
DONE CONFIGURING B	
OPTIONS: Message Level set to 2.	
I2C_CONTROL = 0x04	
Temperature Sensors	
A YES	
B YES	

Q YES Threshold: 80 C Initializing USB: DONE MAIN MENU (Serial Number #0806013) 1) Configure from Main.txt 2) Interactive Configuration Menu 3) Check Configuration Status 4) Select new configuration file 5) List Files on CompactFlash card 6) Dump file on CompactFlash card 7) NA g) Display FPGA Temperatures h) Set Temperature Threshold i) Read IIC register j) write IIC register k) Reset USB ENTER SELECTION:	Main Menu allows control of some limited functions over RS232. All of these functions can be controller from other interfaces, so typically this menu is only used for debugging.

Figure 7 RS232 Output

4.2 Check LED status lights

The DN9200K10PCIE8T has many status LEDs to help the user confirm the status of the configuration process.

Check the power Failure LEDs to confirm that all voltage rails of the DN9200K10PCIE8T are within tolerance. If the voltage of any critical power net on the DN9200K10PCIE8T is too high or too low, the board will be held in reset and at least one of the red LEDs will light. In addition, nothing will work on the board. The LEDs are located along the left edge. Each one is labeled with the voltage that it represents. Normally, all of these LEDs are off. If any of these LEDs light, there is a power problem with the board, and you should contact us. The most common problem that will cause these LEDs to light is a problem with the power supply. More on this topic is later, but for now you can try another supply.

Reset LED. When the board is in reset for any reason, including power failure or pressing the reset button, this LED will light RED. The LED is located above the bank of power fail LEDs, next to the "SYS RESET" button. In most situations a RED LED on the board indicates some sort of failure, and you should know why the LED is on.

Spartan DONE. Check the Spartan FPGA status LED located near the Spartan FPGA. If this LED is not BLUE, there is a serious problem with the board. Nothing on the board will work properly is the Spartan did not configure for some reason. One reason this LED might be off is that a recent firmware update failed. Try re-installing the firmware.

User LEDs. When the Main Reference design of each FPGA is loaded, the FPGAs will blink their Yellow/Red/Green "USER LED"s. These LEDs are connected directly to each of the FPGAs.

CF Activity: When the board is in the process of loading FPGA configuration data from the CompactFlash card, the yellow LED next to the CompactFlash card will flicker.

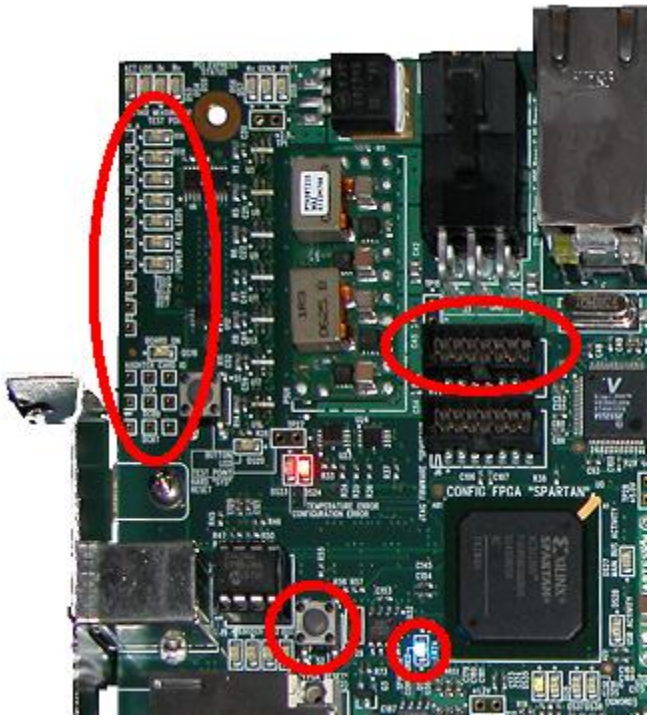


Figure 8: LEDs

5 Run USB Controller

This section will get you started with USB and show you how to operate the provided software.

5.1 Driver Installation

When the DN9200K10PCIE8T powers on, or you connect it to a USB port for the first time, the computer will ask you to install a driver.

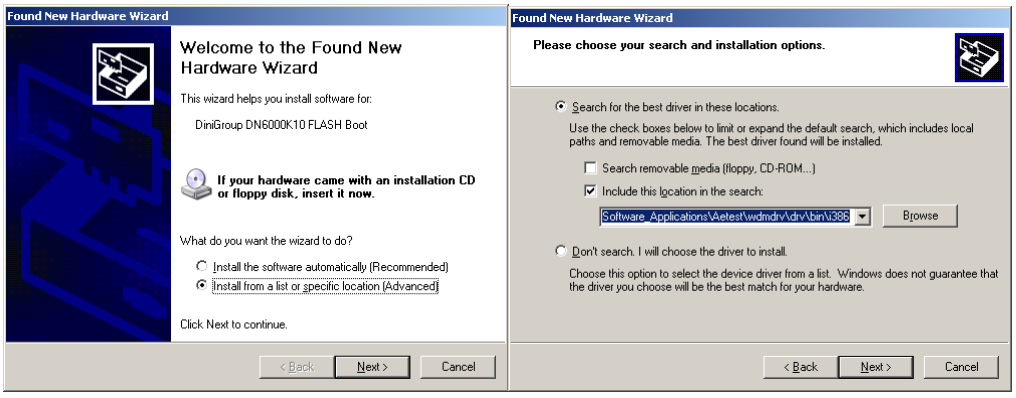

Found New Hardware Wizard
Welcome to the Found New Hardware Wizard
This wizard helps you install software for:
DiniGroup DN6000K10 FLASH Boot
If your hardware came with an installation CD or floppy disk, insert it now.
What do you want the wizard to do?
Install the software automatically (Recommended)
Install from a list or specific location (Advanced)
Click Next to continue.
< Back
Next >
Cancel
Found New Hardware Wizard
Please choose your search and installation options.
Search for the best driver in these locations.
Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.
Search removable media (floppy, CD-ROM...)
Include this location in the search:
Software_Applications\Aetest\wdmdrv\drv\bin\i386
Browse
Don't search. I will choose the driver to install.
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.
< Back
Next >
Cancel

Figure 9 - Driver installation Wizard

In the window that appears, select "Install from a list or specific location". Select Next.

Click "Include this location in the search" and browse to
D:\USB_Software_Applications\driver\windows_wdm
Select Next.

In the next window, select the item in the list "Dini Group ASIC Emulator". Click FINISH.

After Windows installs the driver, you will be able to see the following device in the "ASIC Emulators" group in the Windows device manager: "DiniGroup Product FLASH Boot ".

5.2 Operating the USB Controller program

Run the USB controller application found on the product CD in
D:\USB_Software_Applications\USBController\USBController.exe

Some parts of the program may break if you try to run the program from the User CD without copying it to your hard drive.

DiNi Products USB Controller
File Edit FPGA Configuration FPGA Reference Design Mainbus Settings/Info Production Tests Service
Refresh
Enable USB->FPGA Com.
?
Clear Log
Scroll Log
BOARD TYPE: DN9200k10PCIE8T
USB to FPGA communication is disabled.Enable if you want to use reference design features.

Figure 10: USB Controller Window.

This window will appear showing the current state of the DN9200K10PCIE8T. If FPGA configured, next to each FPGA a blue light will appear.

The window shown above should appear. If the program shows a message box that says, "No devices found", then either the driver is not installed properly, or the computer does not see the device over USB.

5.2.1 Configure an FPGA

Even though the reference design should already be loaded (because you had a Compact Flash card installed when the board powered on), let's configure an FPGA over USB.

To clear an FPGA of its configuration, right-click on an FPGA, and selecting from the popup menu, "Clear FPGA". The blue light above the FPGA on the board, and the virtual blue LED above the FPGA in the GUI should both turn off.

To re-configure that FPGA using the USB Controller program, right-click on the FPGA and select Configure FPGA via USB from the popup menu. The program will open a dialog box for you to select the configuration file to use for configuration. Browse to the provided user's CD "D:\FPGA_Reference_Designs\Programming_Files\DN9200K10PCIE8T\MainRef\LX330\fpga_a.bit"
If you are configuring an LX220 or LX110 device you should select a bit file from the LX220 or LX110 directories instead. Failing to select the correct type of bit file will result in the USB Controller program to warn you, and the FPGA will fail to configure. The program will report the status of the configuration when it finishes. "DONE did not go high". ("DONE" refers to the DONE SelectMap signal, which is asserted by the FPGA when it is properly configured. "DONE" is semantically the same as "is configured")

If you are configuring FPGA B or FPGA Q, you should select fpga_b.bit or fpga_q.bit instead. Should you configure the wrong FPGA with a bitfile intended for another FPGA, the FPGA will succeed to configure, but probably won't function properly (because the pinout are different for each of the six FPGAs). This is not recommended because it could lead to bus contention and excessive heat generation.

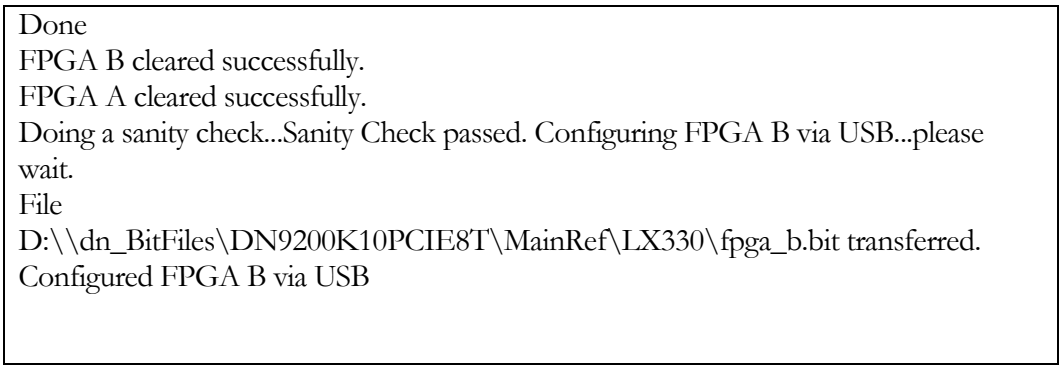

Done
FPGA B cleared successfully.
FPGA A cleared successfully.
Doing a sanity check...Sanity Check passed. Configuring FPGA B via USB...please wait.
File
D:\\dn_BitFiles\DN9200K10PCIE8T\MainRef\LX330\fpga_b.bit transferred.
Configured FPGA B via USB

Figure 11: USB Controller Log Output

The message box below the DN9200K10PCIE8T graphic should display some information about the configuration process. When the configuration is successful, the green LED should re-appear next to the FPGA.

5.2.2 Set Clock Frequencies

The FPGA logic is run on external clocks whose frequencies are generated on the board according to the commands in the main.txt file. Three of these clocks, G0, G1 and G2 can be of whatever frequency the user desires.

To change the clock frequencies of G0, G1 or G2, select the "Clock settings" option from the "Settings" menu.

A dialog box appears asking to which frequency you would like to set each clock. Enter 200, 250, and 200 MHz for G0, G1 and G2 respectively. The Dini Group reference design may only work when the clocks are set within a given frequency range.

5.3 Run Hardware Tests

The provided bit files on the CompactFlash card can be used to interact over USB with the USB Controller program.

Let's run two tests. Make sure the reference design is configured in both FPGAs.

5.3.1 Clock Frequencies

First, hit the "Enable USB->FPGA communication" button. From the "reference design" menu, select "read back clock frequencies". Select any FPGA that is configured. It should print out a list of all clocks connected to that FPGA, along with its frequency, measured from within the FPGA logic.

5.3.2 DDR2

If you do not have DDR2 modules installed in the memory sockets, you might as well skip this step, unless you would like to simulate running the test in a failure condition.

If you haven't already, hit the "Enable USB->FPGA communication" button. This must be done before the program can interact with the reference design.

The DDR2 test requires certain frequencies to be set for it to work without errors. The correct settings are G0: 250MHz, G1: 250MHz, G2: 200MHz.

Additionally, changing clock frequencies while an FPGA design is running can cause errors in the logic. To combat this you will need to reset the logic in the FPGAs. You can do this by pressing the "User Reset" button on the board.

From the FPGA Memory menu, select Test DDR. A box will appear and ask which FPGA should be tested. Select A or B is the correct answer. The log window will report whether the test passed. If it fails, it will print a list of addresses and data that failed.

5.3.3 Other Hardware Tests

This program can somehow be used to test all of the hardware on the board including interconnect and clocks.

5.4 Getting data to and from the FPGA

The USB Controller program also allows you to easily configure and transfer data to and from the user design on the emulation board. This data transfer occurs over the board's "MainBus". This interface is described in the Hardware chapter.

Before USB can be used to operate "MainBus", you must hit the "Enable USB->FPGA communication" button near the top of the USB window.

To read data from the FPGA design (the Dini Group reference design), select from the menu MainBus->Read

In the resulting dialog box, enter "080000000" in the "Start Address" box and "10" in the "Size" box. Press OK, and then DONE. The result of the read is printed to the USB Controller log window.

```
                    --- FPGA READ ---

                    ADDRESS          DATA

                    0x08000000       0xdead5566
                    0x08000001       0x00000000
                    0x08000002       0x05000135
                    0x08000003       0xfffffffe
                    0x08000004       0x34561111
                    0x08000005       0x00000001
                    0x08000006       0x00000000
                    0x08000007       0x00000000
...
```

Figure 12: USB Controller Log Output

The address 0x080000000 is by "MainBus" convention assigned as part of the space available for implementation by FPGA A on the DN9200K10PCIE8T. If FPGA A is not loaded with the Dini Group reference design (or a design that implements the MainBus slave), then all address reads will return 0xDEADDEAD.

Reading from the address 0x18000000 will demonstrate communication with FPGA B.

6 Run AETest_wdm

If you did not install the DN9200K10PCIE8T into a PCI express slot before you powered on your computer, then you will have to skip this step.

The program provided to access the DN9200K10PCIE8T over PCIe is called AETest. It is located on the user CD
D:\PCIe_Software_Applications\Aetest\aetest\aetest_wdm.exe

If you are running Linux or Solaris, you must compile AETest (and driver) before continuing this quick-start guide. This involves installing the kernel source packages on the computer, then loading a kernel module somehow. Details are in the Software Chapter. The rest of this guide assumes you are using Windows XP or Vista.

After you turn your computer on the computer will display a dialog asking for the driver for a "Dini Group board with Virtex 5 PCI Express"

Click "Choose a driver to install" -> Click "Have Disk" and browse to
D:\ PCIe_Software_Applications\Aetest\wdmdrv\drv\dndev.inf

6.1.1 Use AETest

Run AETEST_wdm. The AETest application should display its main menu.

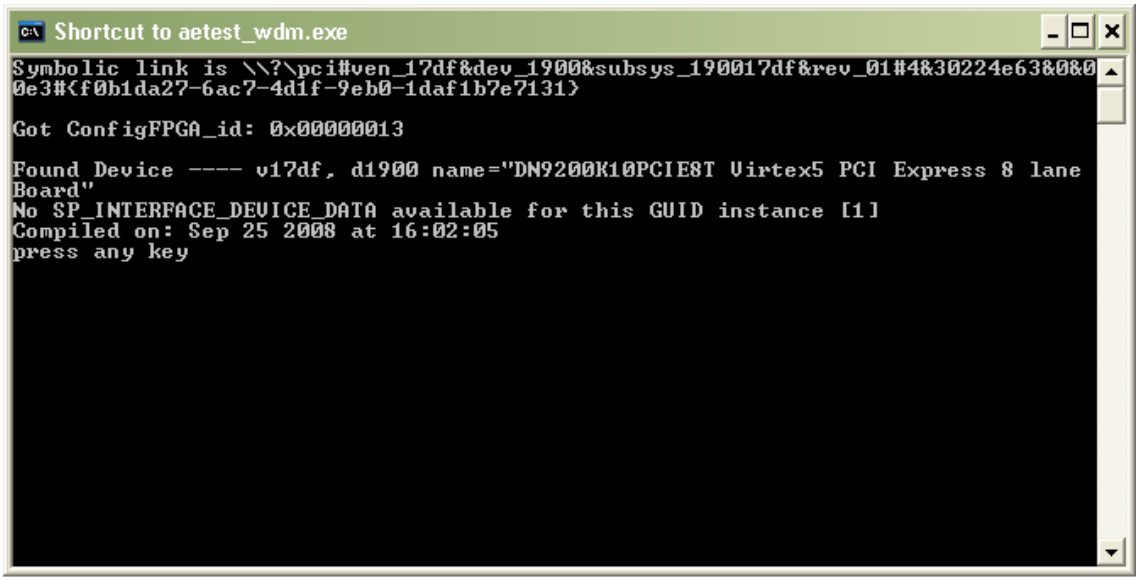

Shortcut to aetest_wdm.exe
Symbolic link is \\?\pci#ven_17df&dev_1900&subsys_190017df&rev_01#4&30224e63&0&0
0e3#{f0b1da27-6ac7-4d1f-9eb0-1daf1b7e7131}
Got ConfigFPGA_id: 0x00000013
Found Device ---- v17df, d1900 name="DN9200K10PCIE8T Virtex5 PCI Express 8 lane
Board"
No SP_INTERFACE_DEVICE_DATA available for this GUID instance [1]
Compiled on: Sep 25 2008 at 16:02:05
press any key

Figure 13 - Splash screen

If this window says something like "GUID not found", then the driver is not installed properly. Check in the windows device manager and see if a device with VID 0x17DF and PID 0x1900 is there.

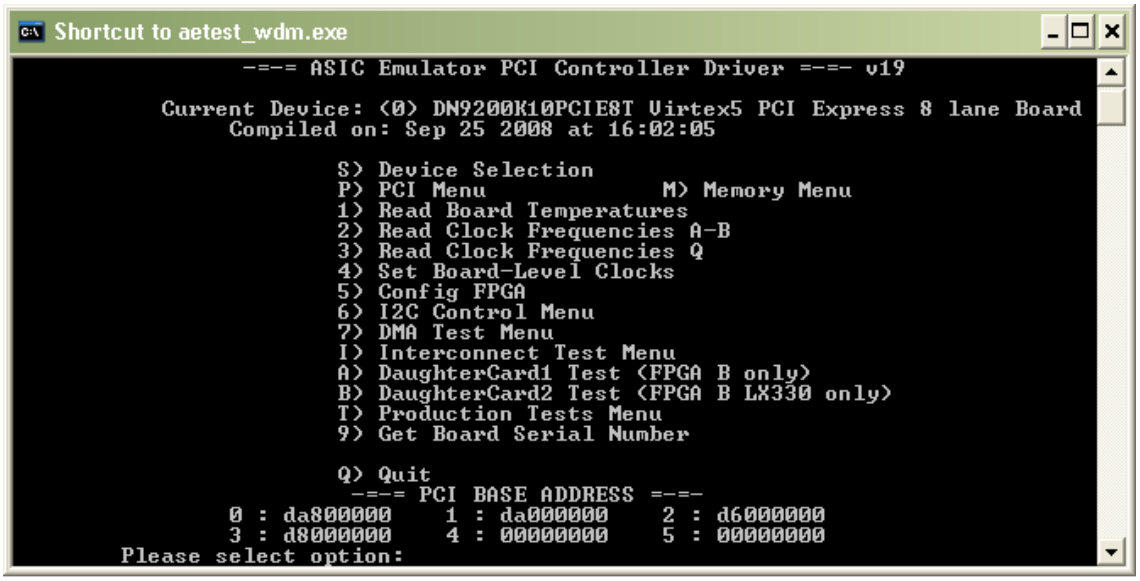

Shortcut to aetest_wdm.exe
--== ASIC Emulator PCI Controller Driver ==-- v19
Current Device: (0) DN9200K10PCIE8T Virtex5 PCI Express 8 lane Board
Compiled on: Sep 25 2008 at 16:02:05
S) Device Selection
P) PCI Menu
M) Memory Menu
1) Read Board Temperatures
2) Read Clock Frequencies A-B
3) Read Clock Frequencies Q
4) Set Board-Level Clocks
5) Config FPGA
6) I2C Control Menu
7) DMA Test Menu
I) Interconnect Test Menu
A) DaughterCard1 Test (FPGA B only)
B) DaughterCard2 Test (FPGA B LX330 only)
T) Production Tests Menu
9) Get Board Serial Number
Q) Quit
--== PCI BASE ADDRESS ==--
0 : da800000
1 : da000000
2 : d6000000
3 : d8000000
4 : 00000000
5 : 00000000
Please select option:

Figure 14 - AETest Main Menu

This is the menu, with some things you can do. To read and write to the user design in the FPGAs, use the "Memory Menu". The "Main Bus" is accessible. This is the same address space that was available to us earlier over USB.

You can additionally access the fast direct PCI Express interface to FPGA A, using the PCI "Bar Read" and "Bar Write" functions. The lowest 4Kb of space in Bar 2 is assigned to a scratch memory residing within FPGA A.

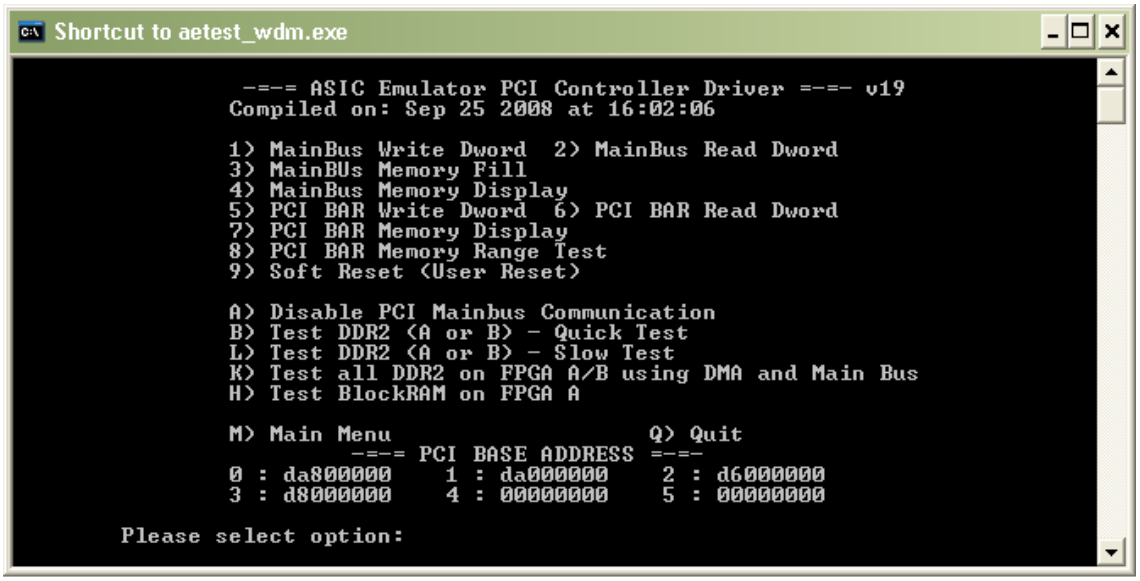

Shortcut to aetest_wdm.exe
--== ASIC Emulator PCI Controller Driver ==-- v19
Compiled on: Sep 25 2008 at 16:02:06
1) MainBus Write Dword
2) MainBus Read Dword
3) MainBUs Memory Fill
4) MainBus Memory Display
5) PCI BAR Write Dword
6) PCI BAR Read Dword
7) PCI BAR Memory Display
8) PCI BAR Memory Range Test
9) Soft Reset (User Reset)
A) Disable PCI Mainbus Communication
B) Test DDR2 (A or B) - Quick Test
L) Test DDR2 (A or B) - Slow Test
K) Test all DDR2 on FPGA A/B using DMA and Main Bus
H) Test BlockRAM on FPGA A
M) Main Menu
Q) Quit
--== PCI BASE ADDRESS ==--
0 : da800000
1 : da000000
2 : d6000000
3 : d8000000
4 : 00000000
5 : 00000000
Please select option:

Figure 15 - Memory Menu

To test high-speed PCI Express access directly to FPGA A (assuming FPGA A is configured), select "PCI BAR Memory Display". Chose bar 0, offset 0. The output of this menu option is

memory on FPGA A. On PCI, when a read result is 0xFFFFFFFF it could indicate a failure. (This is the result returned to software when a hardware timeout occurs on PCI or PCI Express).

It is acceptable to access the DN9200K10PCIE8T from USB and PCIe at the same time. The mutual exclusivity of all features is not finalized, but it's a safe bet that if you use the "MainBus" feature from PCIe and USB simultaneously, the board will do something other than work properly.

7 Scan the JTAG chain

If you wish, you can program the FPGAs using their JTAG interface. Connect a Xilinx Platform USB cable into the FPGA JTAG port (J5), and open the iMPACT program that is installed with Xilinx ISE 10.2.

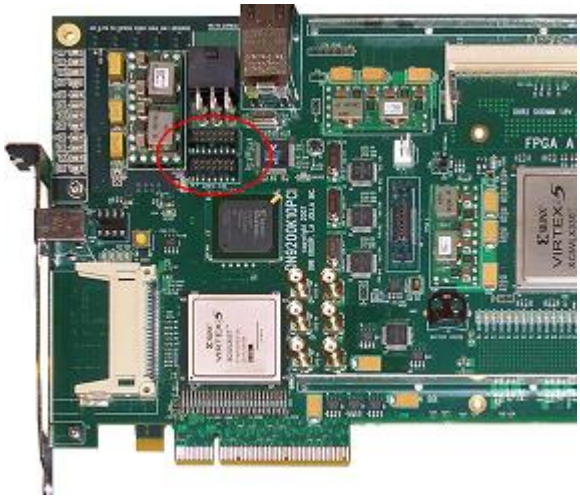


Figure 16 - JTAG Headers

When you connect the Platform USB cable for the first time, Windows will automatically install a driver three times in a row, like a retarded parrot.

The program "scans the chain" to auto-detect the type and number of FPGAs installed on your board and display them on the screen. Right click on an FPGA and select "choose configuration file". Browse to the bit files provided on the user CD. For example:

D:\FPGA_Reference_Designs\Programming_Files\DN9200K10PCIE8T\MainRef\LX330\fpga_A.bit

This JTAG port should also be used for visibility products like Xilinx ChipScope.

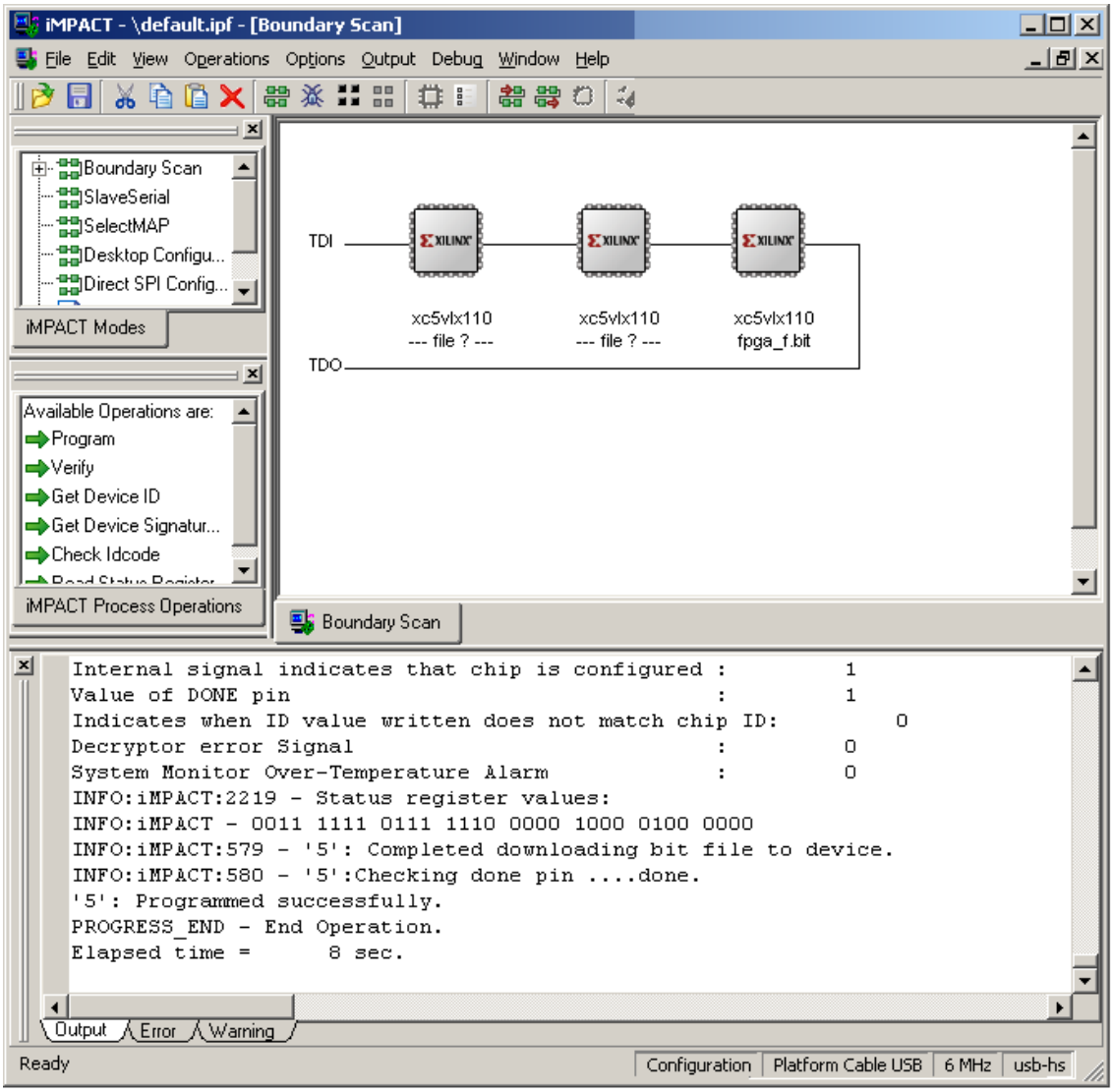


Figure 17 - iMPACT connected to FPGA JTAG

The first item in the chain represents FPGA A, then B, then C and finally at the end of the chain is the PCI Express FPGA (called "Q" by convention).

8 Moving On

Congratulations! You have just programmed the DN9200K10PCIE8T and learned all of the features that you have to know to start your emulation project. Experienced users may want to copy the UCF for the reference design from the user CD into their own projects and never look at the user manual again.

For those new to Xilinx FPGA, the following are suggested starting places:

Using the ISE tool flow, create a bit file that does nothing but routes a clock to an LED, routes reset to an LED, and turns one LED on.

Add a small amount of logic to the reference design.

Read the section describing the external interfaces you wish to use in the hardware section. Find the external interface on the schematic, and the interface chip datasheet on the user CD.

Read the Virtex-5 User Guide, UG200. It can be found in the datasheet directory of the CD.

Chapter 3: Controller Software

The DN9200K10PCIE8T can be hosted from USB or PCI Express. As an example to hosting using these interfaces, the Dini Group provides some controller software that allows configuring FPGAs, and changing the board settings. For more complex host behavior, such as interactively transferring data to and from the board from the host computer, you may have to develop your own host software, either USB or PCIe. At the end of this chapter, there is a programmer's guide to help you interface to the DN9200K10PCIE8T. This, along with the source code of the example software should be able to get you communicating with the DN9200K10PCIE8T.

The software included with the DN9200K10PCIE8T is

USB Controller	A Windows XP or Vista-only GUI application capable of configuring FPGAs, sending data to the user FPGA core via USB, changing board settings, and running hardware tests.
AETest_usb	A cross-platform (Windows, DOS, Linux, Solaris) command-line application capable of configuring FPGAs, sending data to FPGAs via USB, and changing board settings
AETest	A cross-platform (Windows XP, Windows98, DOS, Linux, Solaris) command-line program capable of configuring FPGAs, and sending data to and from user FPGA via PCI Express.

These programs and the source code for them can be found on the user CD
D:\PCI_Software_Applications\Aetest\
D:\USB_Software_Applications\ AETEST_USB\
D:\USB_Software_Applications\USBController\

Precompiled Windows XP binaries for USB Controller, and AETest_usb, and AETest are provided on the user CD as a Microsoft Visual Studio 6 project. Visual Studio 6 or later is required to compile these programs.

All three programs use a driver provided by the Dini Group.

The PCIe drivers can be found at
PCI_Software_Applications\Aetest\wdmdrv
PCI_Software_Applications\Aetest\linuxdrv
PCI_Software_Applications\Aetest\solaris\driver

The USB driver can be found at
USB_Software_Applications\driver

The Linux version of AETest_usb does not require a driver, but does require root access.

1 USB Controller

USB Controller is a GUI program demonstrating the USB capabilities of the DN9200K10PCIE8T. It is compatible with Windows XP and Vista. All capabilities of USB are possible under Linux; however there is no GUI that looks good in these operating systems.

The USB Controller program is intended to
- Verify Configuration Status
- Configure FPGAs over USB
- Configure FPGAs via CompactFlash card
- Clear FPGAs
- Reset FPGAs
- Set Global clocks frequency
- Update firmware (for MCU and Spartan)
- Demonstrate good user interface design practices
- Run hardware tests

1.1 Main Window

The main USB Controller window has the following components: a menu bar, a refresh button, a "Disable USB" button, and board graphic, and a message log. Each item in the menu bar is described later in this section.

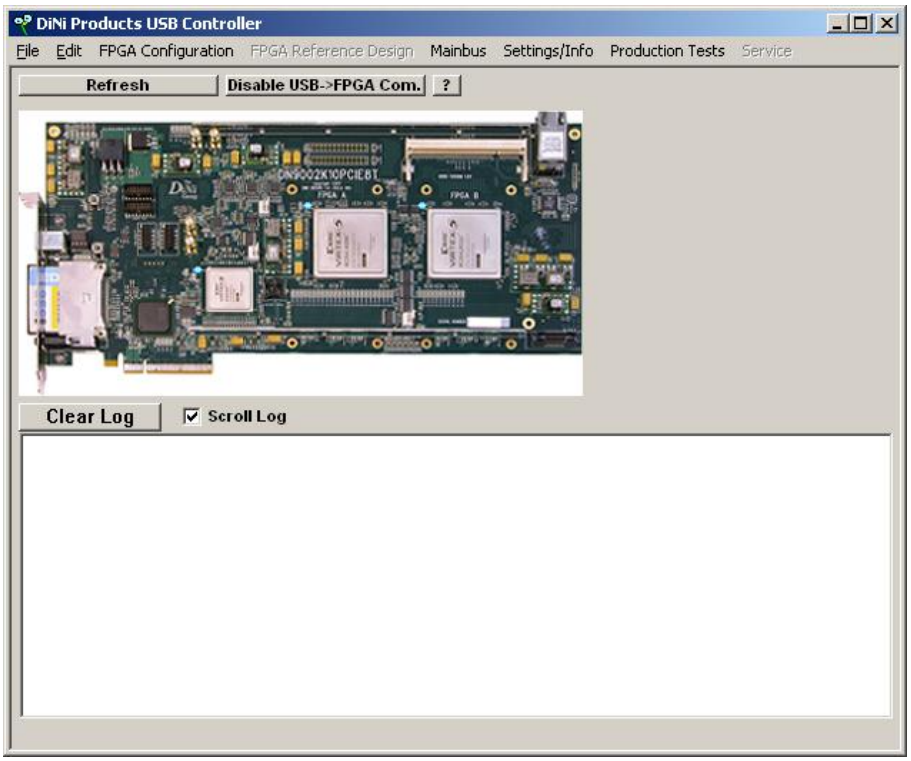


Figure 18 - USB Controller Main Window

1.1.1 Refresh Button

Figure 19 refresh button

The Refresh button updates the board graphic by querying the DN9200K10PCIE8T and reading back its status. The USB Controller program now polls the board constantly, so this button is largely meaningless.

1.1.2 Disable/Enable USB

Figure 20 Enable/Disable button

To communicate to the FPGA design using USB, the "MainBus" interface is used. See the hardware chapter for more information on this interface. Some users elect not to use the Main Bus for USB communication. To allow these users to make use of the signals in the Main Bus for their own purposes, the USB Controller is careful not to use the Main Bus unless explicitly given permission by the user. The user can give permission to use Main Bus by pressing the "Enable USB->FPGA communication" button. It can revoke that permission by pressing the "Disable USB->FPGA communication" button. When the DN9200K10PCIE8T powers on, it begins in the disabled state. The state is stored on the board, so that multiple programs accessing the DN9200K10PCIE8T may prevent each other from using the Main Bus.

1.1.3 Log Window

This text box prints the result of each user command in USB Controller. There is a "clear log" button to clear the contents of this text box.

1.1.4 Board Graphic

USB Controller's main window shows a graphic representing your DN9200K10PCIE8T. The number of FPGAs that are installed on your board should appear in this graphic. If one or more FPGAs are configured on the board, a blue LED will glow next to the FPGA in this graphic window, just exactly like on the actual real board hardware itself.

If the USB Controller could not find a DN9200K10PCIE8T connected to any USB port, this window will appear.

USBController
The DiNi product was not found! Please check the following:
1. Your USB cable is firmly plugged into the computer and the board.
2. Your board is powered on.
3. The device driver for the board is loaded.
4. The device is not presently configuring itself from the media card.
OK

Figure 21 - USB Controller complains if board is not detected

If the board is turned on and plugged in, the USB Controller should be able to detect it. If it does not, try opening the Device manager. You can right-click on the "My computer" icon and select "Hardware tab" and click the "Device Manager" button. This will display a list of the devices connected to your computer. If a Dini Group Logic Emulator appears in the USB section, then USB is working properly on the board, but the program is unable to connect to it. There could be a problem with the driver setup. Select "Switch Device" from the File menu.

If the board does not appear in the Hardware manager, then the DN9200K10PCIE8T may be stuck in reset. See the "Troubleshooting" section in the Hardware chapter. Also, check the red "Reset" LED.

As well as providing visual feedback, the board graphic can be used to control configuration of the FPGAs. To do this, right-click on an FPGA in the graphic to show a contextual menu with the options: Configure, Clear and Reconfigure.

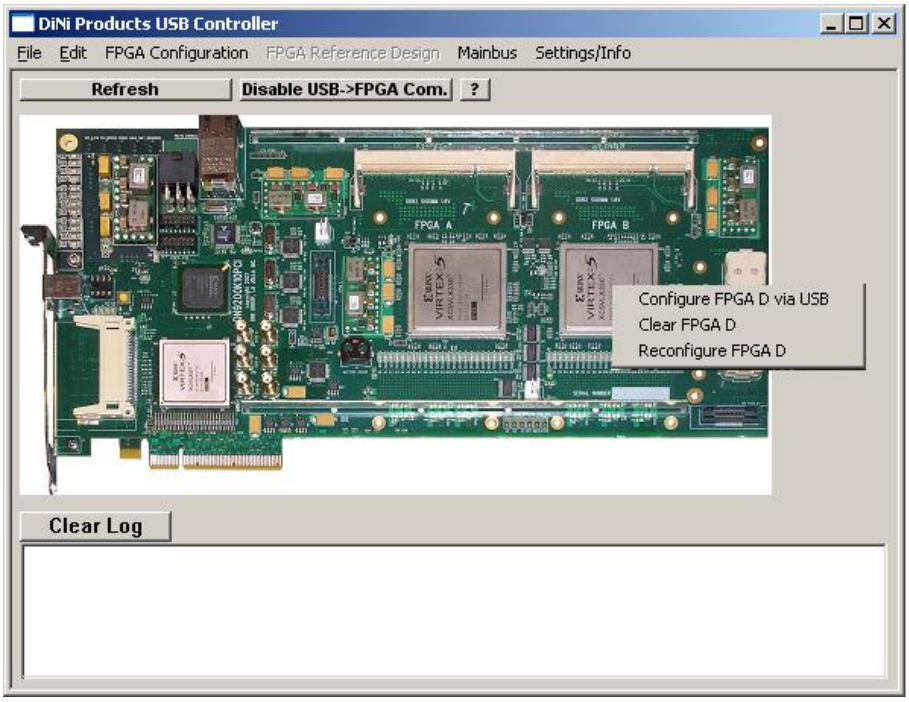


Figure 22 - Configuring FPGAs

Configure will show an Open… dialog for you to select the bit file you wish to use with the FPGA. Clear FPGA will clear and reset the FPGA of its current configuration. Reconfigure FPGA will configure the FPGA with whatever bit file that *this instance* of USB Controller used to successfully configure that FPGA last.

1.2 Menu Options

The following sections describe each menu option and its function.

1.2.1 File Menu

About

Displays USB Controller version number, along with other things.

Switch device

Displays a list of all Dini Group USB devices detects and allows the user to switch the "current" device. The USB Controller will behave as if the "current device" is the only attached Dini Group USB product. Under some situations, the USB Controller may automatically switch device when the "current device" is not valid.

1.2.2 Edit Menu

The Edit Menu performs the basic edit commands on the command log in the bottom half of the USBController window.

Copy, Delete, Select All

1.2.3 FPGA Configuration Menu

The FPGA Configuration Menu has the following options:

Configure Via USB (individual)

This menu option allows you to configure an FPGA. It is equivalent to selecting an FPGA by clicking on it and selecting "Configure", except that this menu option will display a dialog asking which FPGA to configure. Before any FPGA is configured in USB Controller, a "sanity check" is performed. This reads the header out of the binary bit file and determines whether the bit file is compatible with the FPGA installed on the DN9200K10PCIE8T. It will prevent configuration if the "sanity check is not passed" This check can be disabled from the "Settings/Info" menu.

Configure via USB (using file)

This command allows the user to configure more than one FPGA over USB at a time. To use this option you must create a setup file that contains information on which FPGA(s) should be configured and what bitfiles should be used for each FPGA. The syntax of this file is similar or identical to the syntax of the CompactFlash main.txt interface. Details are found in the USB Controller manual on the user CD at

D:\USB_Software_Applications\USBController\doc\USBController_Manual.pdf

Configure via CompactFlash

This command causes the FPGAs to configure based on the instructions in the main.txt file on the CompactFlash card. It will also cause the commands and settings on the main.txt file to be re-issued.

Clear All FPGAs

This command resets all FPGAs, causing them to lose their configuration.

Reconfigure All FPGAs

This menu command is equivalent to selecting "reconfigure FPGA" in the context menu of each of the FPGAs. Each FPGA is cleared before being configured. The last bit file that was loaded via USB for each FPGA is loaded again into the FPGA. If an FPGA has not been loaded with a bit file using *this instance* of USB controller, it is skipped.

Reset
This command asserts the RESET# signal to all FPGAs simultaneously. This is the same signal that is asserted when the user hits the "Soft Reset" (User Reset) button. Its function in the user design is left for the user to define. In the reference design, it causes a global, asynchronous reset. This option also causes the SYS_RSTn signal on the daughtercards to be asserted.

1.2.4 FPGA Reference Design

This menu is not enabled unless "Enable USB" is pressed, and at least one FPGA is configured with the reference design. The USB Controller knows if this is true because it reads a main bus register that is implemented in the reference design. If you compile the reference design yourself, this menu will continue to work as long as you have not removed this main bus register from the design.

Read DDR2 IIC Data
This option will read the contents of the IIC device contained on the DDR2 connected to either of the DDR2 sockets on the board and display them. The reference design automatically configures its DDR2 controller for any DIMM so this feature is more or less useless these days.

Read FPGA Clock Frequencies
This menu option measures and reads back the frequencies of the eight global clock networks, and displays them on the message log. This can help assure you that the clock networks are functioning properly.

1.2.5 Main Bus

The way that user FPGA designs can communicate over USB is the "Main Bus" interface. The "Reference design" menu uses the main bus to read and write registers in the reference design to control the board tests. These tests can be done by the using these menu options without the user having to understand the Main Bus interface or the main bus memory space and its mapping to the reference design. The Main Bus menu allows direct control of the Main Bus. This can be useful if you are using your own FPGA core that implements the main bus.

Write and Read DWORD
This displays a dialog box for writing to the Main Bus address space. It includes some debugging features. All main bus transactions are of length 4 bytes ("DWORD"). The options when using this menu allow the program to automatically read back all written memory locations and compare them to the written bytes. This can be useful when testing a 32-bit memory space.

Test Address Space
This menu option is equivalent to the "Write and Read DWORD" option selecting read, write, use random data, not verbose, show errors. It is much faster. This can be used to test for reliability problems in an address space, for example a DDR memory controller with marginal timing.

Read Address Space to File
This reads data from the main bus at the address specified, and writes the data to a binary file specified. Data on the main bus is in little-endian order. The address after each DWORD is implicitly incremented. (Incrementing behavior can be turned off if a FIFO read behavior is required).

Write Address space from file.
This reads binary data from a file and writes the data to the address on main bus specified. The data is written in little-endian order. The address is implicitly incremented after each DWORD of data. This behavior can be changed to write to a FIFO address (contact support)

Send Command File
This option reads an ASCII file that can contain both reads and writes. Reads will cause the data to be displayed on the log window. The specification for the format of this file is the one which can be inferred from the example below:

AD 08000000
WR 0000FFFF
WR 000000FF
AD 08000000
RD 3

This example writes 0x0000FFFF to address 0x08000000, 0x000000FF to address 0x08000001, then prints out the contents of addresses 0x08000000 through 0x08000002.

1.2.6 Settings/Info Menu

FPGA Stuffing information
Displays a list of the FPGAs on the board, and their type and speed grade. This information is stored in the firmware flash, and is not detected dynamically. You can also get this information off the FPGA JTAG chain (except for speed grade).

Board/Spartan/MCU version
This option is used to read the version number of the current board's firmware. There are two types of firmware, the "Flash" and the "Prom". The two types of firmware, the reference design, and the USB Controller application are only guaranteed to work when using corresponding versions of each. If you update one, you should update the others.

Read FPGA temperatures
Displays the current temperature of the on-die FPGA temperature sensors.

Force Memory Menu display
When the Dini Group reference design is not loaded in at least one FPGA, the "FPGA Reference Design" menu is disabled. This menu command forces that menu to be displayed in this situation. The USB Controller determines if the Dini Group reference design is loaded by

reading a memory location on Main Bus and comparing the result to a predetermined value. This menu may also be disabled because the "USB->FPGA Communication" is disabled.

Turn on Mass Storage Device
This menu option will change the USB behavior of the board so that it appears as a CompactFlash card reader to your computer.

Toggle Sanity Check
normally, the software will prevent the programming of an FPGA with a bit file compiled for any type of FPGA other than the one installed on your board. This menu option will disable this behavior.

FPGA Readback
This menu option will read the entire contents of the FPGA programming memory and write them to a file. The file is a raw binary from the SelectMap bus, so to make any sense out of it, you will have to parse through the binary data.

Hide Board Image
This will make the window much smaller to make use of the USB Controller program easier on small displays, like those on an oscilloscope or iPhone.

Setup clock frequencies
This menu option displays a dialog box allowing the three frequency-selectable global clock networks to be configured.

Global Clock Mux Settings
This allows you to change the frequency source for the clock networks that have a selectable frequency source.

1.2.7 Production Test

Test DDR
This menu option runs a MainBus address range test on the DDR that is selected. This menu item does not configure the FPGA with the reference design, correctly set the clocks or reset the FPGAs. It will fail if these steps are not complete.

One Shot Test
This menu option contains most of the hardware tests that can be run on your board. The tests that this menu run work identically to the hardware tests that your board passes before shipping. There are some options available in the settings dialog window:

-**Main One Shot Test**: contains interconnect, main bus, clock, pull-ups
-**DDRs Test**: Tests DIMMA and DIMMB connections. You must have a DDR2 SODIMM installed in each socket before the test is run.
-**Headers Test**: You should uncheck this box. It will fail without a test fixture.

-**Ethernet Test**: You should uncheck this box. It requires a test fixture.
-**External clocks test**: This test requires a test fixture.
-**Test FPGA Q**. This will test interconnect between FPGA A and FPGA Q
-**LVDS Frequency**: This is the frequency that the FPGA-to-FPGA interconnects will run during the test. 450 is the standard test frequency.
-**Bitfile Path**: This is where the program will get the reference design bit files. They were on the provided user CD.
-**Iterations Count**: The number of consecutive times the entire test will run.

1.2.8 Service Menu

Update Firmware
Update Synthesizer Tables

1.2.9 Debugging Menu

There is pretty much nothing in the debugging menu that you would want to look at except maybe the "Read Configuration Register" and "Write Configuration Register". These menu options read and write to the "Configuration Registers" Described in the "config section" part of the hardware chapter.

1.3 INI File

Some command considered "debugging" commands save persistence information in an "ini" file that gets created in the same directory as the USB Controller executable. This file should not be generated for most users. If it is generated, you can safely delete it, unless you like it. Some of the settings that can be stored in this file are the Text Editor Selection settings, the location of (path to) the reference design programming files (for one-shot-test), and enabling the debug menu.

2 AETest USB

The command line USB controller program is called "AETEST_USB". It provides a subset of the features available on USB Controller and is cross platform. This program is a convenient place to start if you are going to be writing a custom IO controller for USB to communicate with the DN9200K10PCIE8T.

3 PCI Express AETest Application

AETEST utility program can test and verify the functionality of the DN9200K10PCIE8T Logic Emulation board, and provide data transfer to and from the User design.

All AETEST source code is included on the CD-ROM shipped with your DN9200K10PCIE8T Logic Emulation kit. AETEST can be installed on a variety of operating systems, including: Windows 2000/XP/Vista (Windows WDM) and Linux

3.1 Compiling AETest_usb

AETest_usb can be compiled using Microsoft Visual Studio 6 or later, or on any version of Linux that supports the usbdevfs library.

A make file is provided, but you must un-comment one of the following lines to define which operating system you are running. In Windows, you should run nmake.

#DESTOS = WIN_WDM
#DESTOS = LINUX
#DESTOS = SOLARIS

Run nmake on windows and make on linux.

3.1.1 Compiling the Driver

Compiling the driver on windows requires the windows driver development kit. A script "Makeit.bat" can be run from within the windows DDK build environment.

Most people don't need to compile the driver in windows because it already works.

In Linux, the driver must be compiled unless you happen to be using the same architecture and OS version as ours when we compiled it.

3.2 Functionality

All communication to the board using this program is over PCI express. In this way, the basic functionality of PCI Express is tested.

The AETEST utility program contains the following tests:
DMA and BAR accesses over PCI Express (When using the "full function PCI Express endpoint now with DMA™" design for LXT)
DDR2 Memory Test
Flash Test

AETEST also provides the user with the following abilities:
Recognize the DN9200K10PCIE8T
Display Vendor and Device ID
Set PCIe Device and Function Number
Display all configured PCIe devices
Various loops for PCIe device-function and ID numbers
Write and Read Configuration DWORD (for board settings)
Access to the "Main Bus" interface.
BAR Memory operations
Configure/Save BARs from/to a file
Configure FPGAs.

3.3 Running AETEST

The following images show a terminal session in Windows XP.

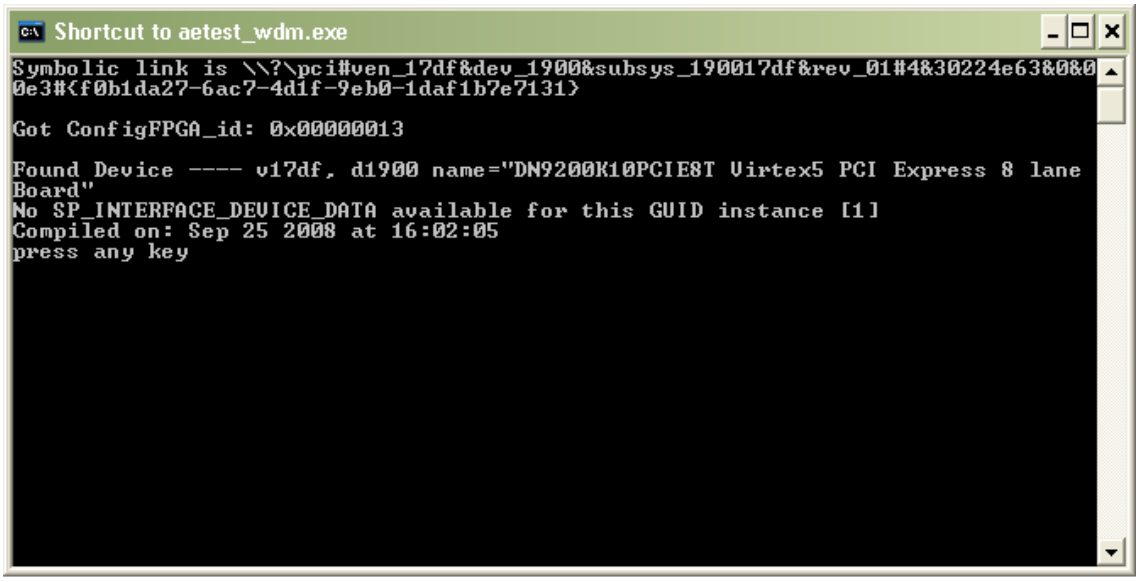

Shortcut to aetest_wdm.exe
Symbolic link is \\?\pci#ven_17df&dev_1900&subsys_190017df&rev_01#4&30224e63&0&0
0e3#{f0b1da27-6ac7-4d1f-9eb0-1daf1b7e7131}
Got ConfigFPGA_id: 0x00000013
Found Device ---- v17df, d1900 name="DN9200K10PCIE8T Virtex5 PCI Express 8 lane
Board"
No SP_INTERFACE_DEVICE_DATA available for this GUID instance [1]
Compiled on: Sep 25 2008 at 16:02:05
press any key

Figure 23 - AETest splash screen

The initial display of AETest shows the results of its scan of the PCIe bus. If the driver for the DN9200K10PCIE8T is not installed, then the software will display a message that no device was found. If this occurs, (and you are using windows), look into the computer's hardware manager and see if a PCI Device with Vendor ID 0x17DF appears. If it does then there is a software or driver problem. If it does not then there is a hardware problem. Look on the board near the 6-pin PCI Express power connector. There is a row of LEDs corresponding to the PCI Express status signals. RED LEDs for LOS indicated the board is not linking with its link partner. Yellow is activity. Three green LEDs a valid link in 1x, 4x or 8x mode respectively.

Below is the main menu.

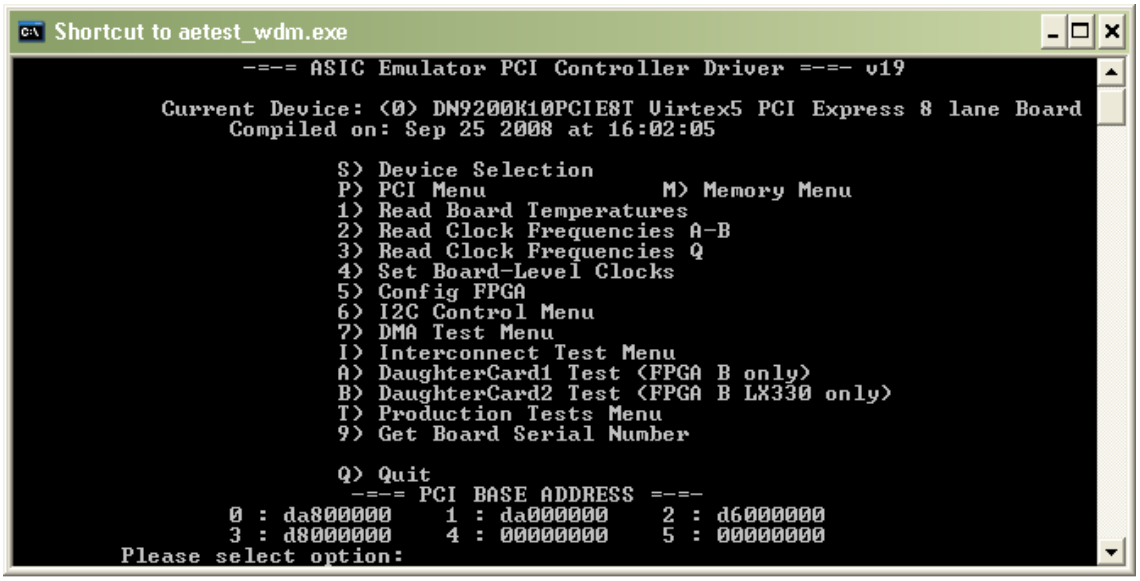

Shortcut to aetest_wdm.exe
-=-= ASIC Emulator PCI Controller Driver =-=- v19
Current Device: (0) DN9200K10PCIE8T Virtex5 PCI Express 8 lane Board
Compiled on: Sep 25 2008 at 16:02:05
S) Device Selection
P) PCI Menu
M) Memory Menu
1) Read Board Temperatures
2) Read Clock Frequencies A-B
3) Read Clock Frequencies Q
4) Set Board-Level Clocks
5) Config FPGA
6) I2C Control Menu
7) DMA Test Menu
I) Interconnect Test Menu
A) DaughterCard1 Test (FPGA B only)
B) DaughterCard2 Test (FPGA B LX330 only)
T) Production Tests Menu
9) Get Board Serial Number
Q) Quit
-=-= PCI BASE ADDRESS =-=-
0 : da800000
1 : da000000
2 : d6000000
3 : d8000000
4 : 00000000
5 : 00000000
Please select option:

Figure 24 - AETest main menu

Below is the PCI menu. It can help you debug a software problem detecting or communicating with the board. The "config DWORD" refers to PCI configuration space, which is normally only controlled by the operating system or BIOS.

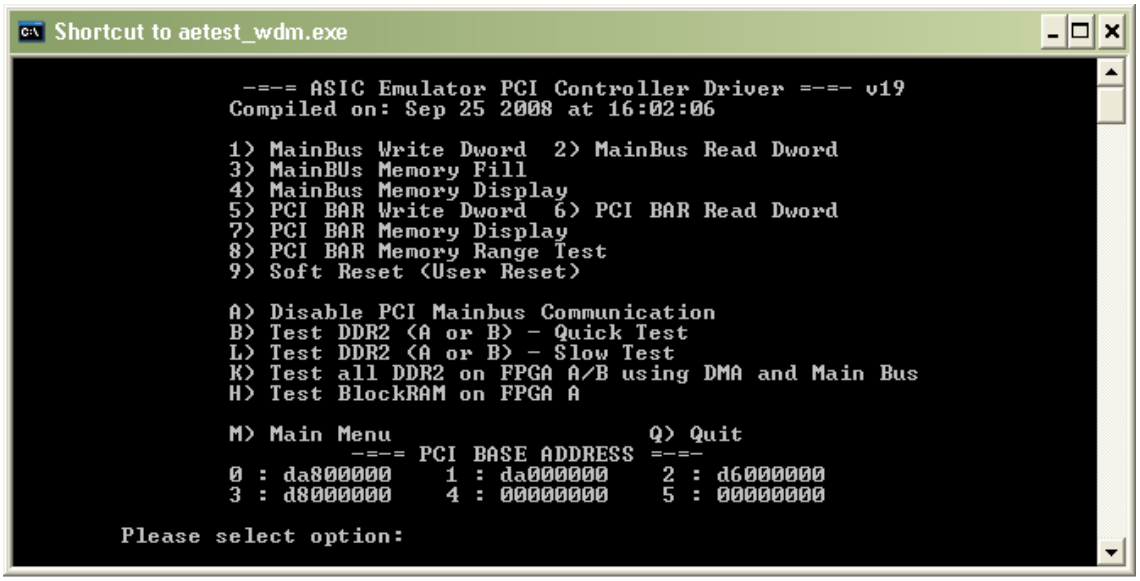

Shortcut to aetest_wdm.exe
-=-= ASIC Emulator PCI Controller Driver =-=- v19
Compiled on: Sep 25 2008 at 16:02:06
1) MainBus Write Dword
2) MainBus Read Dword
3) MainBUs Memory Fill
4) MainBus Memory Display
5) PCI BAR Write Dword
6) PCI BAR Read Dword
7) PCI BAR Memory Display
8) PCI BAR Memory Range Test
9) Soft Reset (User Reset)
A) Disable PCI Mainbus Communication
B) Test DDR2 (A or B) - Quick Test
L) Test DDR2 (A or B) - Slow Test
K) Test all DDR2 on FPGA A/B using DMA and Main Bus
H) Test BlockRAM on FPGA A
M) Main Menu
Q) Quit
-=-= PCI BASE ADDRESS =-=-
0 : da800000
1 : da000000
2 : d6000000
3 : d8000000
4 : 00000000
5 : 00000000
Please select option:

Figure 25 - AETest Memory menu

Below is the memory menu. From here you can communicate with the User design in any of the FPGAs (using Main Bus) or directly to FPGA A. Bar memory and MainBus are different memory spaces.

4 Rolling Your Own Software

Most customers who need to use USB or PCIe as a data interface to their FPGA designs write their own USB and PCIe controller programs, since the USBController and AETest programs do not meet their requirements.

Most of the time, you only need a small change, like for example, you want to read a file off disk and write it to the MainBus interface, blink an LED 4 times, and post the result on Facebook. In this case, let me recommend just modifying the provided AETest or AETest_usb program. These programs are written ~~so that a third-grader could understand them~~ by third graders.

4.1 USB

The behavior of the DN9200K10PCIE8T with respect to a USB interface is given in the Hardware chapter. To access PCI Express from a host software program probably requires a driver. You can use our driver, write your own driver, or try to modify ours.

4.1.1 Windows XP/Vista

BTW: We didn't write this driver. This is the example driver from Cypress provided with the CY7C68013.

When the driver is properly installed in windows, the device will appear as a file in the file system with the following path: "\\\\.\\Ezusb-0".

To interact with the device, open a HANDLE to the device using CreateFile

```
HANDLE handle = CreateFile("\\\\.\\Ezusb-0", GENERIC_WRITE,
                           FILE_SHARE_WRITE, NULL, OPEN_EXISTING,
                           0, NULL);
```

In the case of multiple devices, the paths may be "EzUSB-1", "EzUSB-2", etc.

The functions available using the driver are implemented as "control" operations. Use the DeviceIoControl() function in Windows.h.

4.1.2 Linux

To use USB in Linux, use the provided usbdrvlinux.c file provided on the user CD in AETest_usb/driver

Connecting to the device occurs using the driver's usb_open function.

```
int handle = usb_open(0x1234, 0x1234, 0);
```

usb_devfs provides the functions required to do a vendor request or bulk transfer. These are the only two types of communication required.

4.2 PCIe

The behavior of the DN9200K10PCIE8T with respect to a PCI Express interface is given in the Hardware chapter. To access PCI Express from a host software program probably requires a driver. You can use our driver, write your own driver, or try to modify ours.

4.2.1 Windows Driver Hooks

In Windows, to work with a hardware device, it's driver must be loaded. After this, you can interact with the device using a HANDLE object like a file.

To find a path to the device, use these functions:
`SetupDiGetClassDevs()`
`SetupDiEnumDeviceInterfaces()`
`SetupDiGetDeviceInterfaceDetail()`
You will need to know the device's "GUID" in order to get a list of Dini Group devices on the system. (Otherwise, you will have to get a list of all devices on the system and then filter them). The correct GUID is called `DNDEV_GUID`. The value is defined in a header file "GUIDs.h" in the driver code director.

From the device interface detail, you can get the device path, which can be opened using `CreateFile()`

Once you have a HANDLE object for the device, all operations on the device can be done through "control" operations on the HANDLE. Use the function `DeviceIoControl()`

The available control codes (IOCTL's) available to pass to this function are given in the file Ioctl.h in the driver directory. The ones you will use are

`IOCTL_DNDEV_BAR_READ_U32`
The output buffer should contain
struct { uint32 offset,
uint32 barnum};
The input buffer will be a single uint32. Offset is a byte offset from the BAR specified in barnum.

`IOCTL_DNDEV_BAR_WRITE_U32`
The output buffer should contain
struct { uint32 offset,
uint32 barnum,
uint32 data };
Where offset is a the desired byte offset from the BAR location, barnum is the number of the

BAR that you wish to access, and data is the 32-bit word that you would like to write to the given offset.

4.2.2 Linux Driver Hooks

When the device driver is loaded, the devices will appear on the filesystem at /dev/dndev/

Open the device using open(). The driver implements a hander for the mmap() routine. Therefore, to access PCI Space, you need only to mmap the file to user address space.

Call ioctl using the control code `DNDEV_IOC_GETDEVICE`. This will return an object giving the contents of the base address registers and BAR rangers of the device. When calling mmap() you need to tell the device which BAR you wish to map. This is done by using the offset field of mmap(). When the offset field is somewhere within page 0, BAR0 is mapped. When it is somewhere within page 2, BAR2 is mapped, etc.

```
Void* User_space_pointer =
      mmap(NULL, bar_sizePROT_READ | PROT_WRITE,
            MAP_SHARED,filedes,desired_bar_number*getpagesize());
```

Now PCI Express accesses can be completed by dereferencing *user_space_pointer.

5 Updating the Firmware

Dini Group may release firmware bug fixes or added features to the DN9200K10PCIE8T. If a firmware update is released you will need to download this new code to the firmware flash of the DN9200K10PCIE8T.

There are three firmware files that Dini Group may release.

MCU Flash
The on-board microcontroller controls the configuration of FPGAs, the setting of clocks, USB transactions, temperature sensors, CompactFlash and various other functions. The firmware is stored on a Flash chip.

Spartan Flash
The Spartan "Config" FPGA controls the data paths for Main Bus (PCIe and USB), CompactFlash and some other functions. This FPGA is programmed from a Xilinx configuration PROM. Sometimes, this prom needs to be updated.

PCI Express Flash
If you are using the "Full function PCI Express endpoint now with DMA™" design provided with the board (default), then Dini Group may offer updates and features to this endpoint. The data is stored in an SPI flash which contains the FPGA configuration data for the "FPGA Q" LXT part.

Clock Frequency Tables
This table contains all the PLL settings required to set the Si5326 clock synthesizers. This table will probably never need to be updated.

Stuffing Tables
This table contains a table describing which FPGAs are installed on the board, so the software can act more intelligently. This table probably will not need to be updated ever.

When updating any firmware, the "Flash", "Prom" and USBController.exe should all is updated simultaneously, since Dini Group only tests this code using corresponding versions of each.

5.1 Obtaining the updates

The firmware update files are not posted on the web site. In order to obtain them, you must request them from support@dinigroup.com. You may be required to perform a firmware update to your board to receive support and some features. If a firm ware update is deemed critical to the proper function of the board, a customer notice may be issued.

5.2 Updating the Spartan (PROM) firmware

When updating firmware, you should update in the following order:
1) USB Controller.exe
2) Spartan PROM firmware
3) MCU Flash
4) LTX Bitfile (hex file)

All firmware may have interdependencies, so all four software should be updated at the same time.

5.2.1 Using JTAG cable

This update can be accomplished with the Xilinx JTAG programming program, iMPACT. A Xilinx Platform USB cable ($145) or Xilinx Platform USB Cable II helps updating firmware faster. Or you can update Spartan FPGA using USBController under "Settings/Info"->"Update Spartan" menu. This option takes longer than Xilinx Platform USB cable (about 3-5 min) to complete updating.

Connect a Xilinx Platform USB configuration cable to your computer. When the cable is working properly, but not connected to a JTAG chain, the LED on the cable turns amber. When connected to the DN9200K10PCIE8T, the LED turns green.

Connect the cable to the "Firmware" header, J9

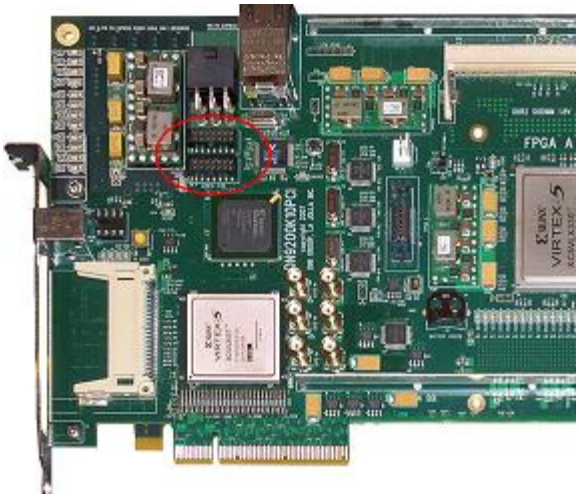


Figure 26 Firmware Update Header

Power on the DN9200K10PCIE8T; When the Platform USB cable is connected to a header, the status light turns green.

Open the Xilinx program iMPACT, usually found at
Start->programs->Xilinx ISE 10.2->Accessories->iMPACT

Choose the menu option File->Initialize Chain. (You may need to create a new project for this menu option to be available)

iMPACT should detect 2 devices in the JTAG chain: xc3s1000 and xc18v04. For each item in the chain iMPACT will direct you to select a programming file for each. For the xc3s1000, press Bypass. iMPACT will then ask for a programming file to program the xc18v04 device. Select the Spartan Firmware update file provided by Dini Group ("prom_flp.mcs"). Hit Open.

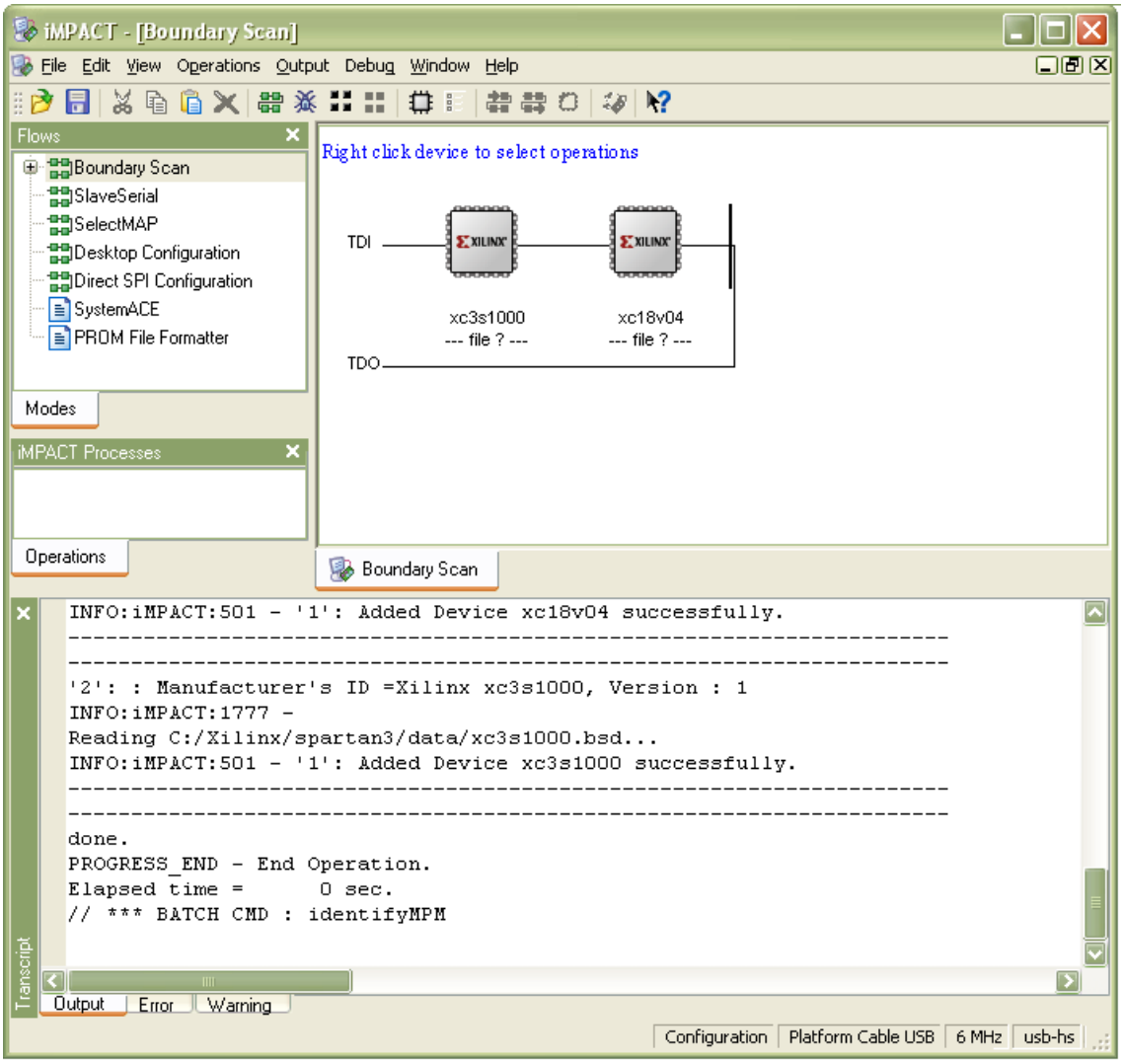


Figure 27 iMPACT Window

To program the prom, right-click on the prom and select "Program…" from the popup menu. In the options dialog that follows, the options "Erase before programming" should be selected, and "Verify" should be selected. Press OK. The programming process should take about 15 seconds over a platform USB cable.

Power cycle the DN9200K10PCIE8T. The new firmware is now loaded. You can close iMPACT and disconnect the Xilinx JTAG cable

5.2.2 Using USBController

If you do not have a JTAG cable, you will need to use the following instructions to update your "Spartan PROM" firmware.

Run USBController.exe. Under "Settings/Info" select "Update Spartan". A warning message will appear to ensure that you want to update Spartan. If you do, hit the "Yes" button. An open file Dialog will appear after that. Please select file "prom_flp.xsvf" provided by The Dini Group. This process will take approximately 75 seconds.

5.2.3 Using AEtest_USB

If you do not have a JTAG cable, you will need to use the following instructions to update your "Spartan PROM" firmware. This update is depending on AEtest_USB and Flash firmware version. Please double check with us (support@dinigroup.com) to make sure that your current version (MCU version, AEtest_USB) supports this option and request *.xsvf file from us.

1. Run aeusb_wdm.exe (or aeusb_linux).
2. At the main menu, please select option 3 "FPGA Configuration Menu"
3. In "Flash Boot Menu", please select option '9'. Note: the option menu is not displayed for security purpose.
4. Please enter the full path filename for the *.xsvf file.
5. Verbose level is '0'. The higher verbose level, the slower the program runs.

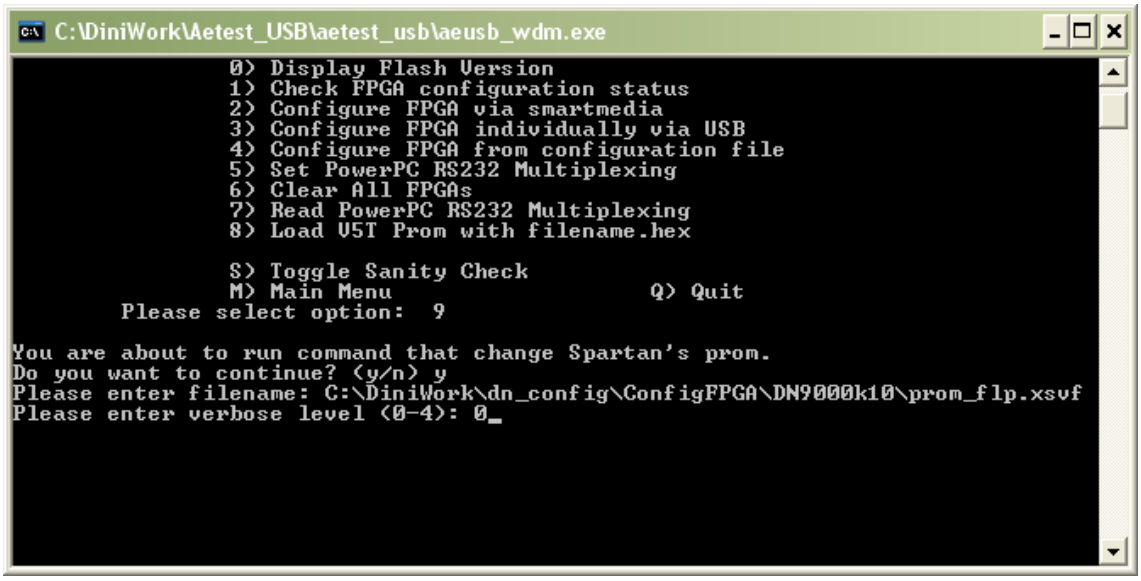

C:\DiniWork\Aetest_USB\aetest_usb\aeusb_wdm.exe
0) Display Flash Version
1) Check FPGA configuration status
2) Configure FPGA via smartmedia
3) Configure FPGA individually via USB
4) Configure FPGA from configuration file
5) Set PowerPC RS232 Multiplexing
6) Clear All FPGAs
7) Read PowerPC RS232 Multiplexing
8) Load V5T Prom with filename.hex
S) Toggle Sanity Check
M) Main Menu
Q) Quit
Please select option: 9
You are about to run command that change Spartan's prom.
Do you want to continue? (y/n) y
Please enter filename: C:\DiniWork\dn_config\ConfigFPGA\DN9000k10\prom_flp.xsvf
Please enter verbose level (0-4): 0

Figure 28 aetest_usb window

6. The progress will start from 0 to 100%. This will take long time to complete (10 minutes). Please do not disturb the process.
7. Power cycle the board when finish.

You can also use commend line: "aeusb_wdm_cmd.exe -XSVF <filename.xsvf>" (or "aeusb_linux_cmd.exe -XSVF <filename.xsvf>").

5.3 Updating the MCU (Flash) firmware

To protect against accidental erasure, the MCU firmware cannot be updated unless the board is put in firmware update mode during power-on. Find Switch S2 ("User Reset") on the DN9200K10PCIE8T.

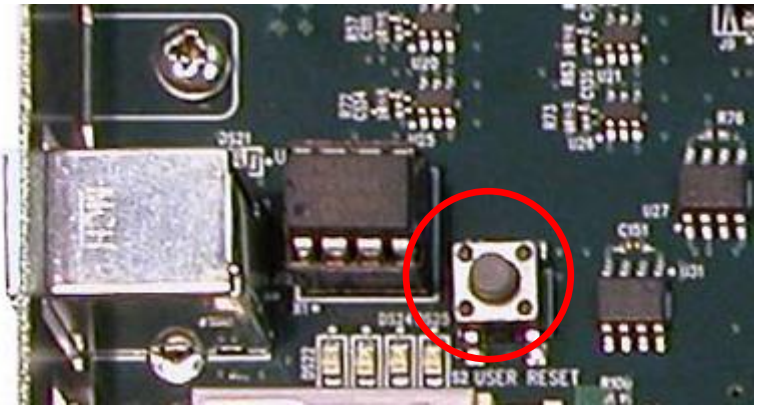


Figure 29 Switch S2

Hold down the "User reset" button while the DN9200K10PCIE8T powers on. Or alternately, while holding down the "User reset" switch, tap the "Hard reset" button. The DN9200K10PCIE8T samples the user-reset button on power on to enter into firmware update mode.

Open the USB Controller program. If the DN9200K10PCIE8T powered on in firmware update mode, there will be dialog boxes, ignore them (press "No") if you not intent to use it. There will be an "Update Flash" button near the top of the USB Controller window. Click on this button.

Figure 30 USB Controller Firmware Update Mode

Do NOT use the "Set FPGA Stuffing" button, as this may cause one or more FPGAs on the board to be inaccessible from the USB Controller program.

When the Open… dialog box appears, navigate to the Firmware image file supplied by Dini Group. The file name should be "firmware.hex". Press OK.

The USB Controller should take about 10 seconds while the firmware update is taking place. A fairly uninformative progress bar should appear while this is happening. When the download is complete, the Log window should print, "Update Complete"

Power cycle the board before doing anything else to make sure the board is no longer in update mode.

You can also use Aeusb_wdm.exe (or aeusb_linux.exe) to update the MCU (flash) firmware. Put the board in the firmware mode and aeusb_wdm. Select option 3 "Firmware Menu" -> option 2 "Update Flash from <firmware>.hex". Enter the ful path filename. It should be firmware.hex that we provide you. The process will take about 2 minutes. When it finishes, please hit "Hard Reset" (S3) on the board or recycle power the board so that DN9200k10PCIE8T can boot from the new User Mode.

5.4 PCI Express Endpoint Firmware

Although the provided configuration files for the LXT "Q" FPGA on your board (responsible to the PCI Express endpoint) are known to be completely perfect in every way, Dini Group may release updates to add features or fix bugs in the PCI Express endpoint. In this case, Dini Group will provide a programming (.hex) file to reprogram the LXT FPGA. This information is stored in an SPI flash device on the board.

5.4.1 Using JTAG USB cable (Xilinx products - iMpact)

To install this updates, plug the USB JTAG cable into the header marked "JTAG FPGA A, B, Q" (J5) on the left edge of the board.

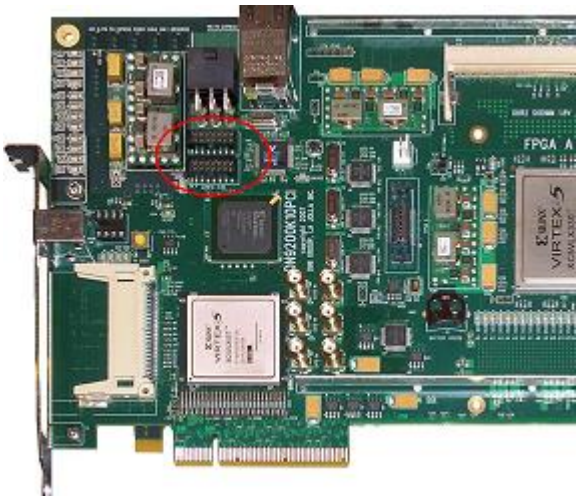


Figure 31 - JTAG Headers

Run iMpact, when you scan the JTAG chain, you will see two user FPGAs of device type LX110, LX220 or LX330. In addition, the last device in the chain will be either a LX50T or FX70T device. Right click on this device and choose "add SPI flash". Then select a firmware file provided by Dini Group that might be called "pcie_v5t.mcs". The program will then for some

reason ask for what type of prom you have. The correct answer is "AT48DB642D". Now, the picture with the six FPGAs will have a little picture of an SPI prom attached to the last FPGA (figure 32). Right click on this, and hit "program". A Box asking about a bunch of programming options will appear. Please check "Erase Before Programming" and uncheck "Verify". Then hit OK. Then wait a little while.

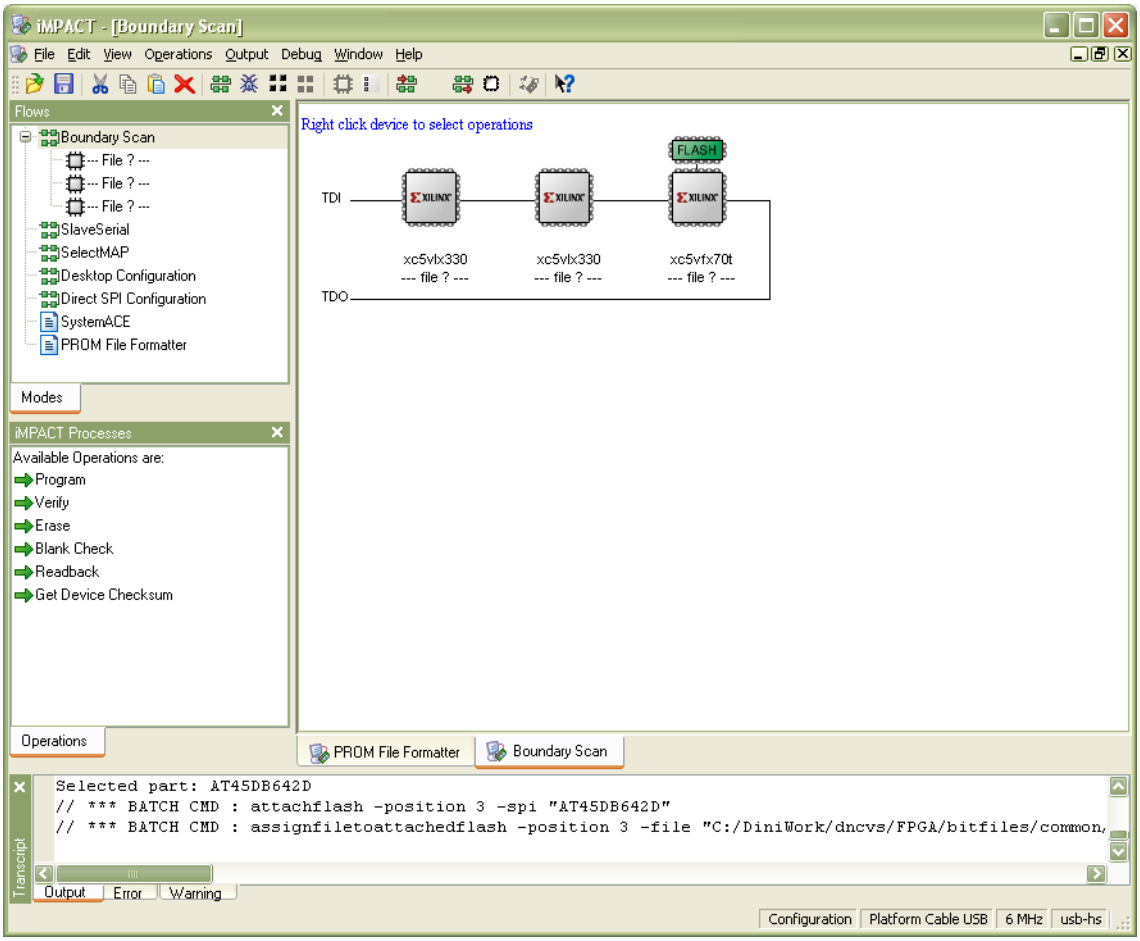


Figure 32 SPI flash is added

Your SPI Flash is programmed.

The SPI prom that is connected to the LX50T (or FX70T) FPGA is where the LX50T (or FX70T) FPGA gets its load file. The LX50T (or FX70T) FPGA can be programmed directly (using a .bit file), but then it will lose its configuration once the board is reset. When you program the SPI flash, it will keep its configuration when the board is reset. A .bit file is used to program an FPGA, a .mcs file is used to program a SPI flash. You can use the Xilinx program iMPACT to generate an .mcs file from a .bit file. The SPI flash can also be updated using USB Controller. When using this method, a .hex file is required.

To generate an .mcs file from a .bit file: in iMPACT, select "generate prom file" and open the provided .bit file. It will ask what the target device is, and it is an SPI Flash of type AT42DB642D. Then double-click generate.

To generate an .hex file from an mcs file. Use the Xilinx program promgen
promgen -w -p hex -r mcsfilename -o outputfilename

5.4.2 Using USBController

You can either generate .hex file from .bit file or contact support@dinigroup.com for new .hex file. Please plug in USB cable and turn the board on.

1. Open USBController.ini and add "service_mode=1". Save and close the USBContrller.ini file

2. Lauch USBController.exe, the Service menu should be selectable.

3. Select "Service"->"ProgramV5TProm", select *.hex file

4. The status bar will be on the bottom of the window. The process takes about 1-2 minutes. Please recycle power the board.

5.4.3 Using AETest_USB

You can either generate .hex file from .bit file or contact support@dinigroup.com for new .hex file. Please plug in USB cable and turn the board on.

1. Run aeusb_wdm.exe (aeusb_linux.exe). Select option '3'
(FPGA Configuration Menu)

2. Select optin '8' (Load V5T Prom with filename.hex), and enter the file name .hex

3. The process takes about 1-2 minutes.

Chapter 4: Hardware

1 General Overview

The DN9200K10PCIE8T ASIC emulation platform is optimized for providing the maximum amount of interconnect between the Virtex-5 FPGAs. It is the lowest cost Virtex-5 FPGA board that has USB, PCI Express and that is exactly 143mm tall.

Below is a block diagram of the DN9200K10PCIE8T.

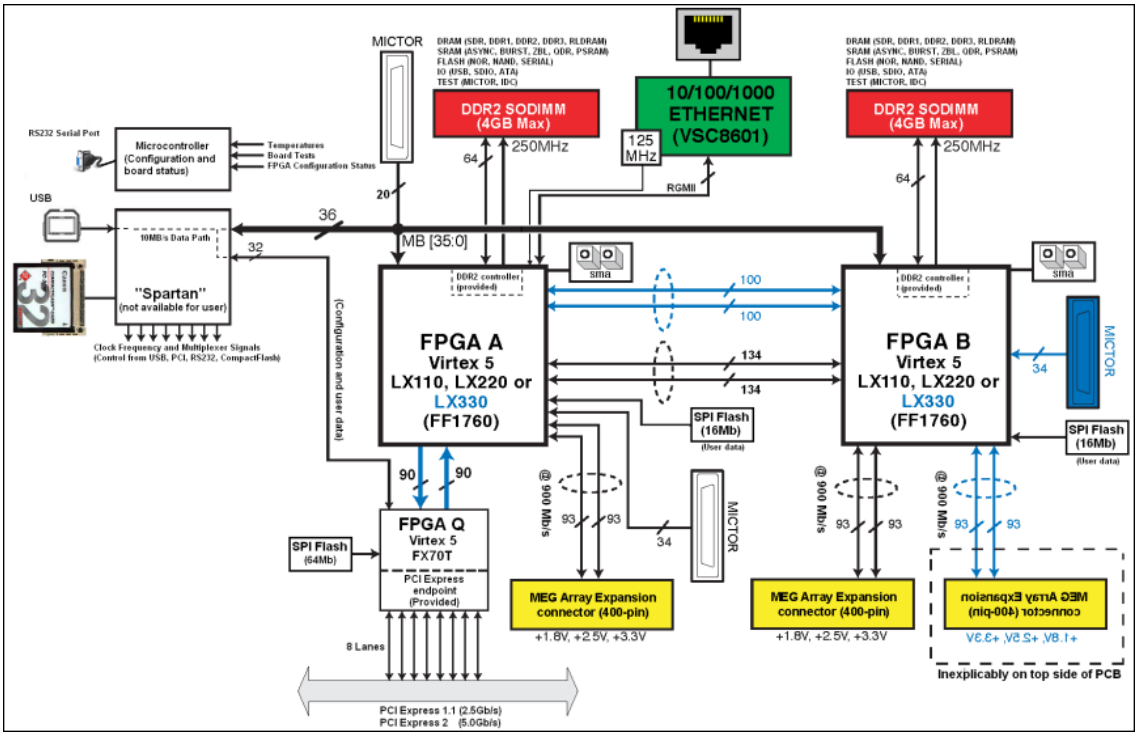


Figure 33 - DN9200K10PCIE8T Block Diagram

The user is expected to implement his external interfaces by designing his own daughtercard to connect to one of the three expansion headers, or hope that Dini Group happens to have a daughtercard or SODIMM card that provides the required external interface.

The board can operate inside a PC as a PCI Express card, or stand-alone on a desk top or swivel chair.

2 Virtex 5 FPGAs

The DN9200K10PCIE8T allows the use of LX110, LX155, LX220 or LX330 FPGAs in each of the positions of FPGA A and B. These FPGAs are in the FF1760 package.

Virtex 5 is the same as Virtex 4, but with a 6 input LUT instead of a 4 input LUT. According to Xilinx, this makes the Virtex 5 30-50% denser and faster than Virtex 4, but it's a lie. Additionally there are some added features over the previous generation of FPGAs, like PLL, ODELAY and "serial transceivers that don't self-destruct after 300 hours of use"™.

2.1 Stuffing options

Either A or B can be left with no FPGA installed to reduce cost. These FPGAs must be in the FF1760 package.

A third FPGA, a Virtex 5 LX50T part is used (as FPGA "Q") for a PCI Express interface. This part is not optional. It will be installed with a LX50T part unless you request a FX70T part instead. An FX70T upgrade is required for Gen 2 PCI Express.

Installing any FPGA other than LX330 for FPGAs A and B impacts the hardware resources available on this board. The block diagrams and feature lists assume LX330 parts.

2.1.1 Q: So Can I get two SX240s?

A: No. It's not a FF1760.

2.1.2 FPGA A and B:

Select an FPGA part to be supplied in each position, A and B. Possible selections are

NONE
LX110 –1 –2 –3
LX155 -1 -2 -3
LX220 –1 –2
LX330 –1 –2

2.1.3 CES Parts

Engineering sample ("CES") parts are no longer offered on this board.

2.1.4 "Small" FPGAs

The DN9200K10PCIE8T is optimized for two Xilinx Virtex-5 LX330 FPGAs. Optionally, it can be ordered with LX110, LX155 or LX220 FPGAs instead. When installed with one or more LX110, LX155 or LX220 FPGAs, the amount of available interconnect is reduced due to the fact the on these parts, some of the package balls have no corresponding IO sites on the chip. A block diagram is given below showing the available resources on the board, where both FPGAs are "Small" type.

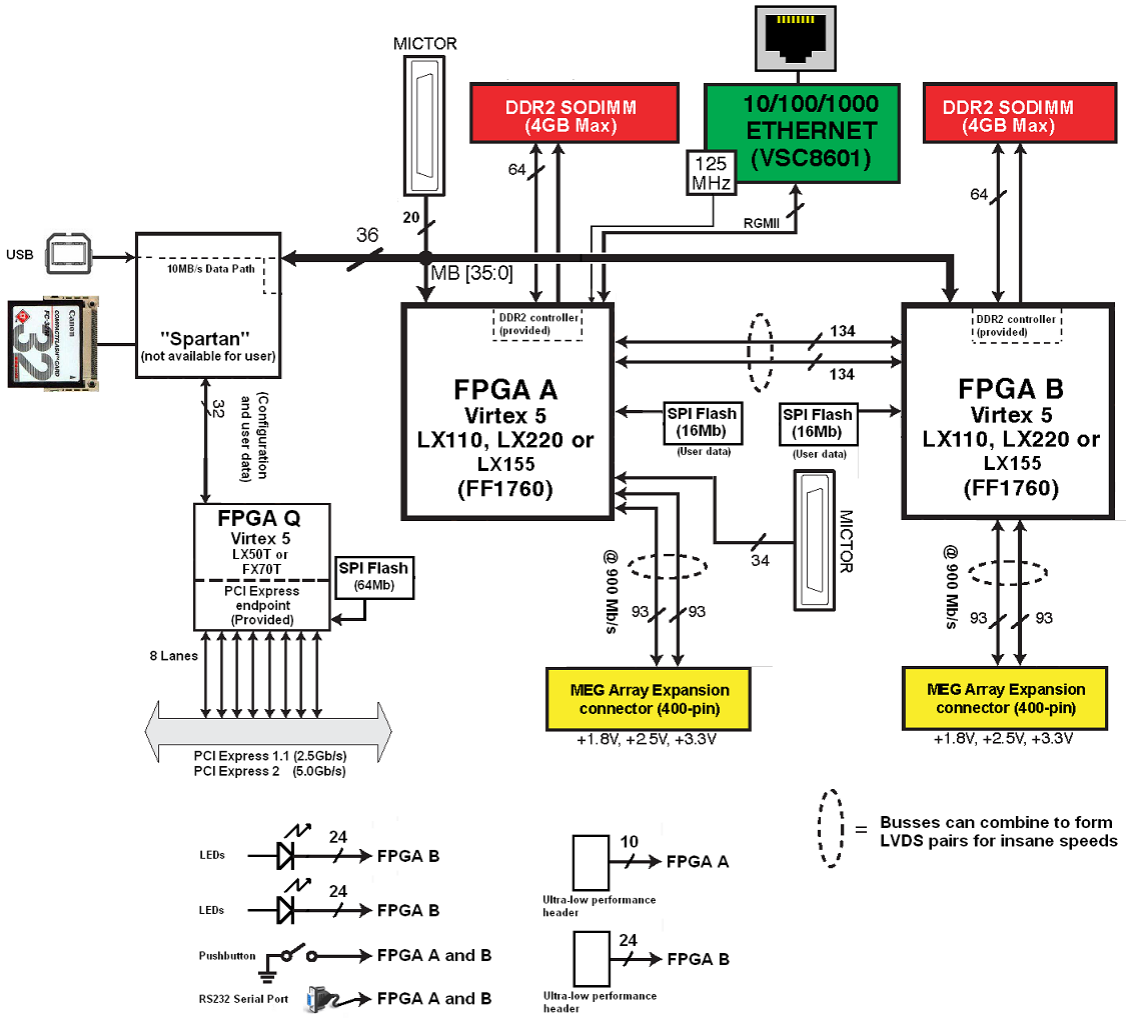


Figure 34 - DN9200K10PCIE8T LX110 Block Diagram

- The amount of interconnect between FPGAs are reduced.
- Daughtercard DCBT is not available.
- FPGA A cannot directly communicate directly with FPGA Q

  Note: PCI Express can still be used for either configuration of FPGAs, or for user data. For user data, the user must use the MainBus interface.

-FPGA B Mictor is not available.

DIMMs, Ethernet, Flash Memory, and MainBus are not affected.

Also, you should analyze your design to determine if the internal resources available in the LX110 and LX220 are sufficient to meet your needs. The FPGA selection guide from Xilinx is printed below.

	Part Number	XC5VLX110	XC5VLX155	XC5VLX220	XC5VLX330
Logic Resources	DSP48E Slices	64	128	128	192
	Slices [2]	17,280	24,320	34,560	51,840
	Logic Cells [3]	110,592	155,648	221,184	331,776
	CLB Flip-Flops	69,120	97,280	138,240	207,360
Memory Resources	Maximum Distributed RAM (Kbits)	1,120	1,640	2,280	3,420
	Block RAM/FIFO w/ECC (36Kbits each)	128	192	192	288
	Total Block RAM (Kbits)	4,608	6,912	6,912	10,368
Clock Resources	Digital Clock Managers (DCM)	12	12	12	12
	Phase Locked Loop (PLL)/PMCD	6	6	6	6
Speed Grades	Commercial	-1, -2, -3	-1, -2, -3	-1, -2	-1, -2
	Industrial	-1, -2	-1, -2	-1, -2	-1

Figure 35 - LX Selection Guide

2.1.5 FPGA Q (PCI Express FPGA) Options

By default, an LX50T FPGA is installed in the FPGA Q position, providing a PCI Express interface for the board. At your request, a different FPGA can be installed here. A list of the available options is given below.

	Part Number	XC5VLX30T	XC5VLX50T	XC5VSX35T	XC5VSX50T	XC5VFX30T	XC5VFX70T
Logic Resources	Slices [2]	4,800	7,200	5,440	8,160	5,120	11,200
	Logic Cells [3]	30,720	46,080	34,816	52,224	32,768	71,680
	CLB Flip-Flops	19,200	28,800	21,760	32,640	20,480	44,800
Memory Resources	Maximum Distributed RAM (Kbits)	320	480	520	780	380	820
	Block RAM/FIFO w/ECC (36Kbits each)	36	60	84	132	68	148
	Total Block RAM (Kbits)	1,296	2,160	3,024	4,752	2,448	5,328
Clock Resources	Digital Clock Managers (DCM)	4	12	4	12	4	12
	Phase Locked Loop (PLL)/PMCD	2	6	2	6	2	6
Embedded [5] Hard IP Resources	DSP48E Slices	32	48	192	288	64	128
	PowerPC® 440 Processor Blocks	—	—	—	—	1	1
	PCI Express Endpoint Blocks	1	1	1	1	1	3
	10/100/1000 Ethernet MAC Blocks	4	4	4	4	4	4
	RocketIO™ GTP Low-Power Transceivers	8	8	8	8	—	—
	RocketIO™ GTX High-Power Transceivers	—	—	—	—	8	8
Speed Grades	Commercial	-1, -2, -3	-1, -2, -3	-1, -2, -3	-1, -2, -3	-1, -2, -3	-1, -2, -3
	Industrial	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2

Figure 36 - LXT FXT Selection Guide

The "PCI Express full-function w/DMA™" bit files are only provided for LX50T and FX70T parts. To use PCI Express generation 2, an FX70T part is required. The available hardware resources on the board external to the FPGA are unchanged. The only difference between these two FPGA options are the internal capabilities of the FPGA.

2.1.5.1 Q: How many gates will I need?

A: You have to run a design through ISE to get an estimate. You can get a rough estimate by counting the number of flip-flops in your design and using the above selection charts. Always allow for a 40% increase in required area. If you have any minimum frequency requirements, then assume you will only be able to achieve 60% utilization in the FPGA. If you have high fan-outs (average above 5 or 6), then you will only be able to achieve 60% utilization.

2.1.6 Speed Grades

The interface performance characterizations included in this manual and in advertisements are valid for all shipped FPGAs, regardless of speed grade. These numbers are characterizations, and not guaranteed minimum operating conditions.

Therefore, the requirement for higher speed grade parts comes only from the requirements of your design. Before you buy a board, you might want to run a test place and route on the design in Xilinx ISE so that you can see how easily timing can be met in a slower FPGA.

For FPGA Q, the PCI Express FPGA, we will provide the minimum speed grade part required for our provided "full-function PCI Express endpoint now with DMA™" design.

Some interfaces may run at increased speeds above and beyond Dini Group's advertised performances when used with –2 or –3 speed grade parts. Xilinx advertises FPGA-to-FPGA interconnect performance up to 1.2 GHz and DDR2 performance up to 667 MHz. We've never tried.

2.2 Using IO

You must use the provided UCF for the LOC constraint of each pin and the correct IO standard.

2.2.1 Timing

For all interfaces described in this section, the responsibility for meeting IO timing and correctly implementing the physical interface is the users responsibility. For your convenience, a use model is provided for many interfaces where timing is guaranteed by the hardware. Typically, to get the best IO performance from the FPGA, the user will use a DCM in the FPGA to compensate the delay of the internal clock network. When using this method, the timing parameters for the FPGA are given below:

Clock-to-out time:	3.37 ns
Input-to-clock time (setup):	1.0 ns
Clock-to-input time (hold):	0 ns

Higher speed grade parts may have improved performance. If additional performance is required, there are two possibilities:

-Use and external clock feedback path for the DCM. This will reduce clock-to-out time to about zero, but may also cause a non-zero hold time.

-Use a DCM to dynamically adjust the output and input phases of the clocks. This will allow a maximum operating frequency of 500 MHz to 900 MHz, depending on the IO skew. This method is required also on interfaces where there is significant clock skew between the FPGA and external device (like daughter cards or DDR2 SODIMMs).

Always use the minimum IO timing constraints in the UCF because these constraints will prevent flip-flops from getting moved outside of the IO block.

2.3 Hardware Errata Details

There are no errata for Virtex-5 production (non- CES) parts.

2.4 Upgrade Policy

2.4.1 Upgrading to new board

2.4.2 Adding FPGAs to a DN9200K10PCIE8T

Prices are not cost-prohibitive. Call or email sales@dinigroup.com for a quote. Note that there is a physical limit to the number of FPGAs that can be added to your board because the board and FPGAs have a limited number of solder cycles allowed.

3 PCB

3.1 Trace delay

The delay of some signals is given in the user guide. This is additive delay, that is, it should be added to the clock-to-out time provided by the Xilinx tool during place-and-route. For example, if a signal has a trace delay of 0.5ns and the clock-to-out time of an output in your UCF is 3.4ns, then the signal will not be an output high at the receiver pin until 3.9ns after the clock edge.

These numbers are only valid if the outputs are using a correct IO methodology, usually requiring match-impedance outputs (DCI), or terminated receivers. All signals on the board are matched to 50Ω.

Trace delays are only valid on signals from a single source with a single receiver.

3.2 Signal Quality

The maximum noise possible on any user IO signal on the board is about 0.5V

4 Configuration Section

The circuit on the board controlling the FPGA configuration signals is called the "configuration section". It is built around a Spartan 3 FPGA. This FPGA controls the bus on the FPGA that control the FPGA's internal configuration SRAM memory ("SelectMap"). Access to this bus is provided to CompactFlash, USB, and PCI Express.

MainBus is also controlled through this FPGA, but details on using MainBus are given in some other section.

This circuit also has secondary functions:
- Temperature Sensors
- Clock Frequency Control
- Clock Frequency Source Control
- Blink Activity LEDs
- Voltage Monitoring

Some housekeeping functions are performed by a Microcontroller (IDE and USB initialization, serial port).

The configuration data for the Spartan ("Prom") and the code for the microprocessor ("flash" and "Eprom") are collectively known as the "firmware".

Most technical details about the configuration circuit are omitted from this manual, since the user should not require it.

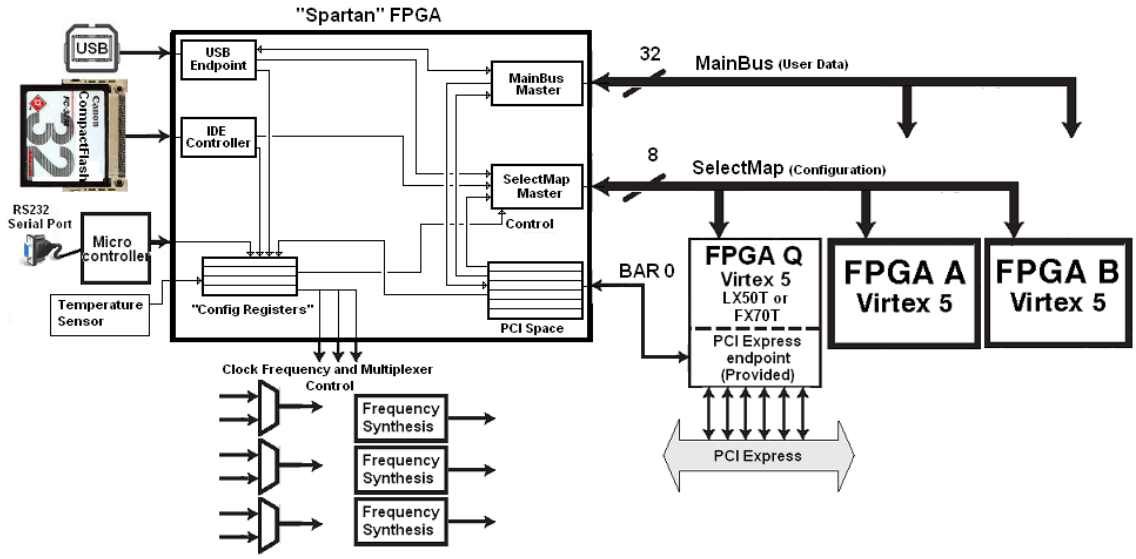


Figure 37 - Config Section Block Diagram

Above it a block diagram of the configuration circuit. Access to the SelectMap and MainBus interfaces are available to USB, CompactFlash and PCI. The "Config Registers" are also available and required to control the SelectMap interface fully.

4.1 Configuration Section Feedback

During normal operation, and in error situations, the configuration section prints messages to the RS232 terminal header (P3). Some very limited functions are also able to be controlled from this interface. See the RS232 output for instructions. These functions include settings the clocks, controlling the process of configuring from CompactFlash, and temperature sensor controls.

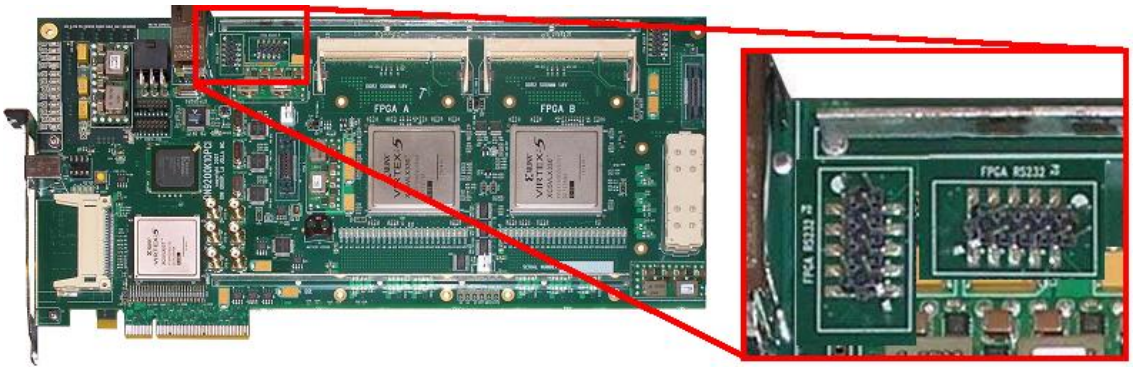


Figure 38 - Serial Port Headers

The configuration section RS232 terminal header, labeled "MCU" above, can be connected to a computer serial port, using the settings:

19200 Baud
No flow control
One stop bits
No parity

The syntax and content of the output messages changes are not given because they change rapidly. This interface is not at all fun to use, and is intended mostly for Dini Group to debug hardware or software failures.

If you need RS232 for your FPGA design, this is not the correct header to use.

4.2 FPGA Configuration

Normally, configuration of the Virtex-5 FPGA occurs over the Virtex-5 "SelectMap" interface. The only configuration method possible on the DN9200K10PCIE8T that does not use this interface is JTAG. For a description of the SelectMap interface, see the Virtex-5 configuration guide.

Typically, the user will supply a "bit" file generated by ISE, and put it on a CompactFlash card, or supply it to software over PCI Express or USB, and the user does not have to understand the SelectMap interface.

USB, CompactFlash and PCIe configuration occur over the SelectMap bus. The configuration section makes no modification of the "bit stream" sent to it over PCIe or USB. It only copies the data to the SelectMap interface. The "bit stream" must contain all of the SelectMap commands necessary to configure and startup the FPGA. These SelectMap commands are created automatically by Xilinx tool bitgen (part of ISE). Not all of the bitstream generation options available in bitgen are compatible with the DN9200K10PCIE8T.

Currently, before configuring the FPGA using any method (except JTAG), the configuration section asserts the PROG# signal of the FPGA to clear it. For this reason, the "disable SelectMap" option in bitgen has no effect.

On each FPGA, the DONE signal is connected to a blue LED located next to each FPGA. This signal gives a quick indication of whether each FPGA is configured or not.

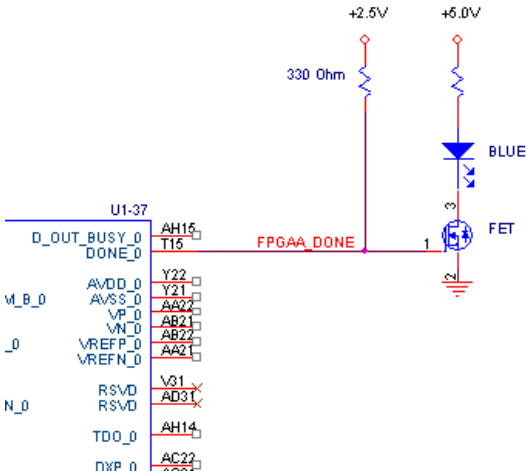


Figure 39 - DONE LED circuit

The data signals, D[7-0] are dual-purpose signals and can be used as additional interconnect pins after all FPGAs have been configured. Care must be taken that the FPGA design does not drive these signals until after all FPGAs have been configured. The configuration section will assert the FPGA_RESET# signal until this occurs (CompactFlash configuration only).

If you use the SelectMap data signals as interconnect, interfacing to the board using USB or PCI may interfere with your design, unless the software is careful. Certainly the provided programs USB Controller, AETEST, and AETest_USB were not written with this possibility in mind.

If using these signals as interconnect, the appropriate drive standard is LVCMOS25. The IO voltage is 2.5V

SelectMap Readback is possible on the DN9200K10PCIE8T. This can be accomplished over PCIe or USB. In order to complete readback over USB, a vendor request is sent to select "readback mode" on one of the USB endpoints, and to automatically send a sequence of SelectMap commands to the FPGA.

The Virtex-5 JTAG configuration method does not go through the configuration circuit. See the JTAG interface section for details about this.

4.3 PCI Express

PCI Express access to the configuration circuit is only available when the provided "full-function PCI Express endpoint now with DMA™" bit files are used in FPGA Q. When a user design or the PIPE design is used, the controls in this section are not available.

In the "full-function" design, BAR0 is reserved for configuration functions. Within BAR 0, offsets below 0x200 are contained within the endpoint's internal registers, and the offsets above 0x200 represent registers within the Spartan FPGA.

4.3.1 BAR0 Map (LO)

Bar0 (LO) are registers contained within the LXT FPGA. The primary use is to control DMA functions. Code to implement DMA using the design is found in the AETest driver directory. The addresses are byte offsets from the BAR0 location. All registers are 32-bit and should not be written or read using byte enables.

0x00	VERSION	Version number for the "full function PCI Express endpoint"
0x04	DATE	Compile data of the "full function PCI Express endpoint"
0x08	DESIGN_TYPE	Constant value
0x0C	GTPCLK_SYNTH	IIC Control of the GTP refclk synthesizer
0x10	RESET_CTRL	
0x14	RS232_CTRL	Turns on and off the RS232 RX and TX signals
0x18	LED_CTRL	Allow manual control of the status LEDs
0x1C	FAN_TACH	Counter connected to the fan tachometer input
0x20	DESC_DMA0_A0	DMA control
0x24	DESC_DMA0_A1	"
0x28	DESC_DMA0_AMASK	"
0x2C	DESC_DMA0_CTRL	"
0x30	DESC_DMA0_POLLI	"
0x34	DESC_DMA0_CURRARD	"
0x38	DESC_DMA0_CURRAEX	"
0x3C	DESC_DMA0_FIFO_COUNT	"
0x40	DESC_DMA1_A0	"
0x44	DESC_DMA1_A1	"
0x48	DESC_DMA1_AMASK	"
0x4C	DESC_DMA1_CTRL	"
0x50	DESC_DMA1_POLLI	"
0x54	DESC_DMA1_CURRARD	"
0x58	DESC_DMA1_CURRAEX	"
0x5C	DESC_DMA1_FIFO_COUNT	"
0x60	CLK_CNT_DMA	Clock counter
0x64	CLK_CNT_USER	Clock counter
0x68	CLK_CNT_CONFIG	Clock counter
0x6C	CLK_CNT_MB48Q	Clock counter
0x70	CLK_CNT_REFQ	Clock counter
0x74	CLK_CNT_GTPQ	Clock counter
0x78	CLK_CNT_EXT0_Q	Clock counter
0x7C	CLK_CNT_EXT1_Q	Clock counter
0x80	CLK_CNT_G0_Q	Clock counter

0x84	CLK_CNT_G1_Q	Clock counter
0x88	CLK_CNT_G2_Q	Clock counter
0x8C	CLK_CNT_TP_Q	Clock counter
0x90	CLK_CNT_MB	Clock counter
0x94	CLK_CNT_CFG	Clock counter
0x98	INTERRUPT	Read/clear interrupt flags
0x9C	INTERRUPT_MASK	Interrupt enable/disable
0xA0	RS232_TOGGLE_CTRL	Enable/Disable RS232

4.3.2 BAR0 Map (HI)

These registers are contained in the Spartan 3 FPGA. Addresses are offsets from the BAR0 location. All registers are 32-bit and should not be written to or read using byte enables.

0x200	DMA_WR_CNT_ADDR	Do not use.
0x208	CONFIG_CONTROL	"Selects" FPGAs. Returns config status
0x210	CONFIG_DATA	Sends one byte of data to SelectMap
0x218	MCU_CLOCK_CONTROL	Do not use.
0x238	FPGA_STUFFING	Array Says which FPGAs are installed
0x240	FPGA_ADDR	Set "current" MainBus address
0x248	FPGA_WRITE	Send word to MainBus
0x250	FPGA_READ	Get word from MainBus
0x258	MCU_WRITE	Write to "Config Registers"
0x260	MCU_READ	Do not use.
0x268	MCU_READ_2	Read from "Config Registers"
0x270	MB_CONTROL	Turn on or off MainBus Auto-Increment

4.3.3 FPGA Configuration

To configure and FPGA over PCI Express follow the steps below. Remember that all BAR0 registers are 32-bit "word" registers (byte-writes have undefined behavior). Addresses are all offsets from the BAR 0 address.

1) "Select" an FPGA.
Address 0x208 is the "Config Control" Register. Its bits [3:0] "select" an FPGA, and bit 4 controls the "Selected" FPGA's PROGn signal.

Write 0x00000011to "select" FPGA A or 0x00000012 to "select" FPGA B.

2) Reset the selected FPGA ("Assert PROGn").
Write 0x00000001 to "prog" FPGA A or 0x00000002 to "prog" FPGA B.

3) Read the current initialization state of the selected FPGA.
When read, address 0x208 will return the SelectMap status signals. Bits [3:0] give the "selected" FPGA, bit 5 is the "PROGn" state, bit 6 is the "INITn" state, bit 7 is the "DONE" state.

After you have set "prog" on an FPGA, poll 0x208 and wait for the "INITn" state to go low (0), to show that it is in reset.

4) Release PROGn
Write a bit 1 to the PROGn of the "Config Control" register. (Use a mask so as not to change the "selected" FPGA

5) Poll INITn to wait for the device to be released from reset.

6) Bang configuration bytes into CONFIG_DATA
CONFIG_DATA register is at address 0x210. Write one byte at a time into the low bits of this 32-bit register. Use bytes directly from the configuration file generated by bitgen. This byte stream contains SelectMap commands and data.

7) Bang junk. Continue banging bytes onto CONFIG_DATA. This is not required is your .bit file already contains enough bytes to account for whatever you startup sequence requires.

8) Poll DONE
Read from address 0x208 and wait for the DONE bit to be high.

9) De-select the FPGA (optional)
Write a 0 to address 0x208 to select "no FPGA"

4.3.4 Readback

This is possible, but not implemented over PCI Express. You can either use the USB readback, or yell at us until we implement over PCI as well.

4.4 Clock Control

4.4.1 Synthesizer Frequencies

The networks that are sourced from Synthesizers (CLK_G0, CLK_G1, CLK_G2) can have their frequencies set over CompactFlash, USB or PCI Express. In order to set the frequency of these clocks, write to the appropriate "Configuration Registers". To correctly use configuration registers of PCI Express, USB, or CompactFlash, see the section on configuration registers.

To set the frequency of G0, first decompose the desired frequency into its whole number and fractional parts. Encode the whole number part in Binary. Encode the fractional part as parts in 1000. Then encode this as a binary number. Write the low 8 bits of the whole number into the register G0_INTEGER_B0 and the rest into register G0_INTEGER_B1. Write the low 8 bits of the fractional part into G0_FRACTIONAL_B0 and the rest into G0_FRACTIONAL_B1. Finally, write a bit into the register PENDING_CLKS to indicate which frequency should be updated. 0x01 is G0, 0x02 is G1 and 0x04 is G2.

To set G1 or G2 use different registers.

Example: Set G2 to 233.75 MHz.
233 in binary is 0xE9
0.75 is 750 parts in 1000.
750 in binary is 2EE.

Write 0xE9 to G2_INTEGER_B0 (0xDFC8)
Write 0x00 to G2_INTEGER_B1 (0xDFC9)
Write 0xEE to G2_FRACTIONAL_B0 (0xDFCA)
Write 0x02 to G2_FRACTIONAL_B1 (0xDFCB)
Write 0x04 to PENDING_CLKS (0xDF40)

4.4.2 Clock Sources

The networks EXT0 and EXT1 can have their PLL frequencies set, their divider values set, their frequency source set from USB, CompactFlash, or PCI Express.

The control of these devices is via bits in a two Configuration Registers, SYNTH_EXT0_CTRL and SYNTH_EXT1_CTRL.

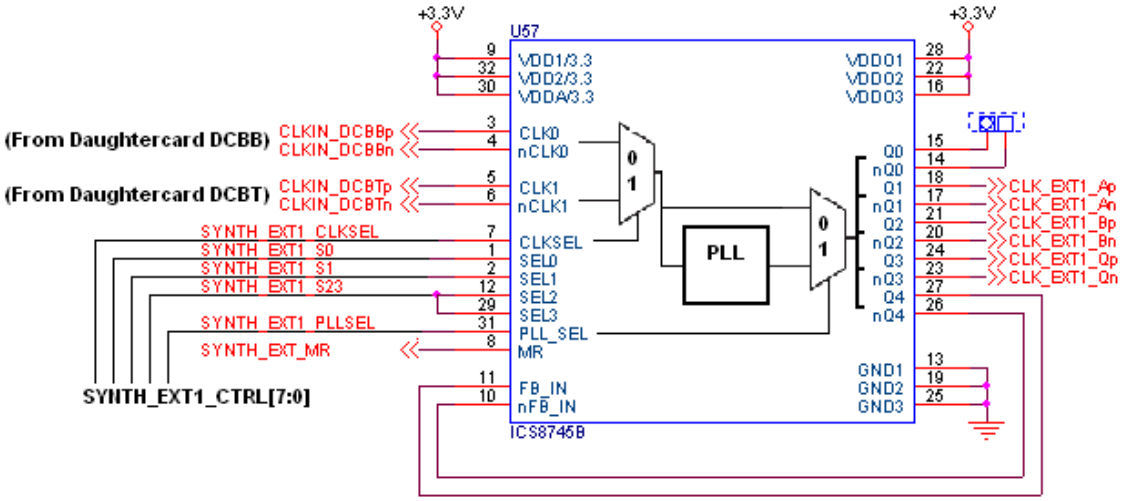


Figure 40 - EXT0 EXT1 Circuit

For operation of the ICS8745B, see the provided datasheet. Register bit 0 controls the CLKSEL signal, bit 1 is the PLLSEL signal, bit 2 is S0, bit 3 is signal S1, and bit 4 is both signals SEL2 and SEL3.

Example: Set CLK_EXT0 to the SMA input (input 1) and bypass the PLL.
Write 0x1E to SYNTH_EXT0_CTRL (0xDF24)

4.5 CompactFlash Interface

Most important settings on the DN9200K10PCIE8T can be controller through the Compact Flash interface. This interface can also be used to configure FPGAs. The CompactFlash interface is not under the direct control of the user, but is accessed only by the configuration logic.

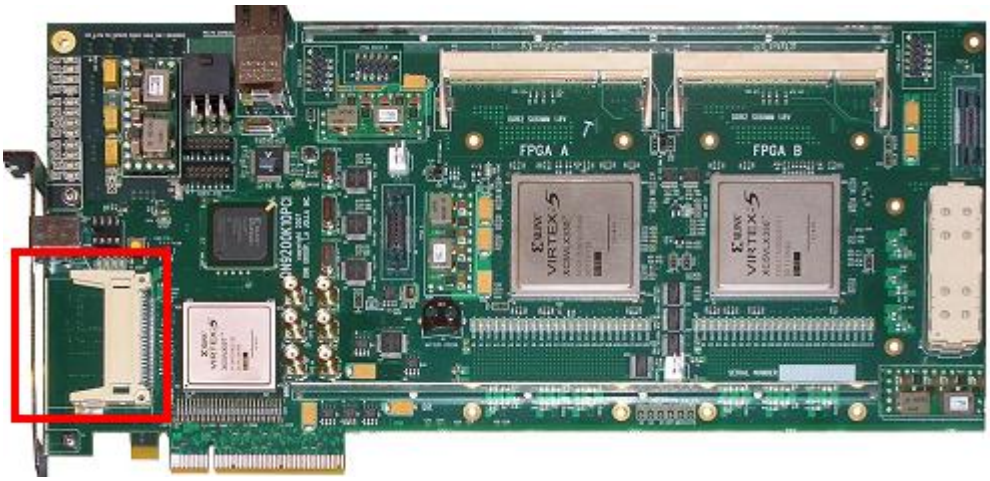


Figure 41 - CompactFlash card socket

The CompactFlash interface can take any sort of CompactFlash card that we know of. If you find one that doesn't work, email it to us and we can add support. The slot is hot-swappable.

In order to make the board configure from the card, you can:
- Reset the board by power-cycling it, or by pressing "Sys Reset" button
- Use the "MCU" RS232 menu option
- Use the USB Controller program (or USB Vendor request)

4.5.1 Main.txt

On the CompactFlash card, you should place a text file with the filename "Main.txt". When the board powers on, it will read this file to determine what to do. You can:

-Configure FPGAs
-Set clock frequencies
-Write to MainBus
-Write to "configuration registers"

A main.txt file contains a list of commands, separated by newline characters. A list of valid main.txt commands is given below.

```
// <comment>
FPGA <FPGA NAME>:<filename>
CLOCK FREQUENCY: G0 <number> [MHz]
CLOCK FREQUENCY: G1 <number> [MHz]
CLOCK FREQUENCY: G2 <number> [MHz]
SOURCE: G0 2
SOURCE: G1 2
SOURCE: G2 2
SANITY CHECK: <yn>
VERBOSE LEVEL: <level>
MEMORY MAPPED: 0x<SHORTADDR> 0x<BYTE>
MAIN BUS 0x<WORDADDR> 0x<WORDDATA>
FILE TRANSFER:
DCLK: DC0 250MHz
```

<comment> can be any string of characters except for newline.
<fpga name> can be one of these: A, B, C, D, E or F
<filename> can be the name of a file on the root directory of the CompactFlash Card.
<number> can be any positive number in decimal. Decimal points are allowed.
<yn> can be the letter y or the letter n
<level> can be 0, 1, 2 or 3
<SHORTADDR> is a 4-digit number in hexadecimal (16 bits)
<BYTE> is a 2-digit number in hexadecimal (8 bits)
<WORDADDR> 8-digit (32 bit) number in hexadecimal representing a main bus address
<WORDDATA> 8-digit (32 bit) number in hexadecimal containing data for a main bus transaction

The following table describes the function of each of the available main.txt commands.

Instruction	Function
// <comment>	The configuration circuitry performs no operation and moves to the next command.
VERBOSE LEVEL: <level>	This command will set the amount of output that will be produced over the RS232 port during configuration. When level is set to 0, the port will produce only error output.
FPGA A:<filename>	The Virtex 5 FPGA "A" will be configured with the file named by <filename>
FPGA B:<filename>	The Virtex 5 FPGA "B" will be configured with the file named by

	<filename>
SANITY CHECK: <yn>	If <yn> is set to y, then the MCU will examine the headers in the .bit files on the CompactFlash card before using them to configure each FPGA. If the target FPGA annotated in the .bit file header is not the same type as the FPGA the MCU detects on the board, it will reject the file and flash the error LED. Before this command is executed, <yn> is set to the default value y. If you want to encrypt of compress your bit files, you will need to set <yn> to n.
MAIN BUS 0x<WORDADDR> 0x<WORDDATA>	Writes data in <WORDDATA> to the address on the main bus interface at <WORDADDR>. This command only makes sense in the context of the Dini Group reference design, unless your design implements a compatible controller on the main bus pins. The Specification for this interface is in MainBus section
MEMORY MAPPED: 0x<SHORTADDR> 0x<BYTE>	Writes to a configuration Register. This command can be used to access features that do not have a main.txt command. Example applications include setting clock sources, settings the EXT0 or EXT1 clock buffers to zero-delay mode, or setting the clocks to frequencies lower than 31MHz.
SOURCE: G0 2 SOURCE: G1 2 SOURCE: G2 2	The SOURCE instructions cause the global clock networks to output a clock from an alternate source. When source of G0 is set to "2", then the global clock G0 becomes a step clock, which can be accessed through config register 0xDF23. When source of G1 is set to "2", the global clock network G1 becomes a step clock which can be toggled by writing to config register 0xDF23. When Source of G2 is set to "2", then the source of the G2 clock network becomes FPGA A, using the "FBACLK" signal.
CLOCK FREQUENCY: <clockname> <number> MHz	The MCU will adjust the clock synthesizer producing clock <clockname> to the frequency <number>.

Figure 42 Main.txt Commands

An example main.txt file is given below.

```
FPGA A: fpga_a.bit
FPGA B: fpga_b.bit
FPGA C: fpga_c.bit
FPGA D: fpga_d.bit
FPGA E: fpga_e.bit
FPGA F: fpga_f.bit

clock frequency: G0 200MHz
clock frequency: G1 250MHz
clock frequency: G2 200MHz
```

Even if you are not planning to configure your Virtex 5 FPGAs using a CompactFlash card, you may want to leave a CompactFlash card in the socket to automatically program your global clock. (Clocks may also be programmed using the provided USB application, or over the PCI Express bus.)

4.5.2 Unimportant CompactFlash Hardware Notes

The Compact Flash interface is hot-swappable.

An activity LED, DS148, located next to the Compact Flash slot indicates activity on this interface.

Please contact support@dinigroup.com if you find an incompatible card, so that we can add software support for it.

Also, the board only accepts CompactFlash cards formatted in the FAT file system. Most new compact flash cards come pre-formatted with the FAT32 file system. In this case, the DN9200K10PCIE8T will not be able to recognize files on the card.

4.6 USB

The USB and PCI Express interfaces can be used for both configuration (FPGA configuration, and clock settings, etc.) or for direct communication with the user design in the FPGA. These interfaces are described individually in their own sections in the hardware chapter.

4.6.1 Configuring an FPGA

The following procedure is used by software on the host computer to configure an FPGA over USB. This procedure is followed by the USBController program and AETest_usb program on the user CD.

1) USB Software gets a handle to a USB device with VID 0x1234 PID 0x1234.

2) USB host software sends vendor request VR_SETUP_CONFIG 0xB7 (see *Vendor Requests*) with 1 byte in the data buffer representing which FPGA to configure. (A is 0x01, B is 0x02, C is 0x03…)

3) The configuration circuit on receiving this vendor request asserts the PROG signal of the selected FPGA. This resets the FPGA and clears any configuration data it may already have. This Vendor request also selects the FPGA, so that SelectMap bus activity only affects the selected FPGA. Bulk transfers initiated after this command to endpoint 2 are interpreted as SelectMap transfers, rather than Main Bus transfers (See *Main Bus access* above). This will be so until vendor request VR_SETUP_END (0xBD) is called.

4) USB host software sends a bulk write USB request to EP2. Each byte of data in the bulk write is sent to the selected FPGA over the SelectMap bus, and the FPGA signal CCLK is pulsed once for each byte of data sent. Note that the LSBit in the USB transaction is sent to the LSBit in the SelectMap interface, so bit swapping as described in the Virtex 5 Configuration Guide is not required. A standard .bit file from Xilinx bitgen can be transferred in binary over this USB interface to correctly configure an FPGA on the DN9200K10PCIE8T. Make sure CCLK is selected as the startup clock in the bitgen settings. This is the default setting.

5) After an FPGA configures, the DONE signal will go high, lighting the blue LED next to the FPGA (labeled "DONE").

6) The USB Controller sends a vendor request out VR_SETUP_END (0xBD). This request deselects the FPGA, so that further bulk requests are interpreted as Main Bus transactions.

4.6.2 Readback

Readback is performed in the same way that configuration, except that the direction of the bulk transfer is BULK_READ instead of BULK_WRITE.

Reading from this endpoint causes one CCLK cycle on the SelectMap bus of the selected FPGA. In order to initiate readback, you must send a vendor request to put the endpoint in readback mode, and send a vendor request that will initiate a SelectMap sequence that puts the FPGA SelectMap bus into read mode.

The data returned from the endpoint is the raw data from the SelectMap bus. In order to make any sense of this data, you will have to muck through the binary data and match it up with the read back, mask, register location list, bitfile, files that were produced by bit gen. Also note that the first few thousand bits are junk, as described in the Xilinx Virtex 5 configuration user guide.

In order to get register state data from the readback stream, you will have to implement the ICAP module in your Verilog. This might mean having a controlled clock with a breakpoint and a trigger condition and a lot of other things that we haven't thought about because nobody seems to care about readback.

4.7 Configuring the "PCI Express" FPGA

All the files that mentioned below are located from the user CD: D:\FPGA_Reference_Designs\Programming_Files\pcie_fpga\pcie_dma folder. Depend on what type of FPGA, you can select which folder LX50T or FX70T. To configure the "pci express" fpga (also referred to as "V5T", "FPGA Q", "LX50T", "MAX", "John's FPGA", "mishap"), there are several methods:

1) Configure from compact flash card. Add a line to the main.txt file:
   FPGA Q: bitfilename.bit
   The next time the board power off and on, this programming data will remain in the card and program the FPGA again.

2) Load image directly over JTAG. Using a Xilinx JTAG cable, connect to the "FPGA JTAG" connector on the board. The last item on the JTAG chain is the "PCI Express FPGA". Right-click on this device and select a bit file. Program the device. The next time the board powers on, this programming data will be lost.

3) Load image into prom over JTAG. Using a Xilinx JTAG cable, connect to the "FPGA JTAG" connector on the board. The last item on the JTAG chain is the "PCI Express FPGA". Right-click on this device (in iMPACT), and select "add SPI flash…" select the image file (a .mcs file). Program the SPI device attached to the FPGA. The next time the board powers on, this image will automatically load into the FPGA.

4) Load the image into the prom over USB. Using the windows USB controller program, you can select from the "Service" menu, "program V5T flash". From the open… dialog, you can select a .hex file. After a minute, the program will load the hex file into the prom. When you power cycle the board, then the programming data will be loaded into the FPGA Q.

5) Load image directly into FPGA over USB. In the windows program USB Controller, you can right-click on the FPGA Q and select "program this fpga". After selecting a .bit file, the program will load the FPGA. When the board powers down and back on, the programming data will be lost.

6) Program the PROM over PCI Express. This isn't very reliable.

PCI Express cannot be used to program the FPGA directly because the FPGA is required to be configured for PCI Express to function.

4.8 Configuration Registers

Some of the controls on the board, specifically the clocks, are accessed though the "configuration registers". PCI Express, CompactFlash and USB all have access to these registers somehow. See the corresponding section.

FPGA Configuration Registers

FPGA_SELECT	0xDF0C	"Selects" an FPGA for the SelectMap interface
FPGAQ_CONTROL	0xDFB0	Allows access to the MSEL pins of FPGA Q
BEGIN_READBACK	0xDFDD	sends a command sequence on SelectMap
END_READBACK	0xDFDE	Sends a command sequence on SelectMap

MainBus Control Registers

PCI_COMMUNICATION	0xDF15	Switches MainBus between PCI and USB mode
FPGA_COMMUNICATION	0xDF39	Disables MainBus (for use as interconnect)
GPIF_EP2TC0	0xDFA0	Maintain Read/Write ordering on MainBus
GPIF_EP2TC1	0xDFA1	"
GPIF_EP2TC2	0xDFA2	"
GPIF_EP2TC3	0xDFA3	"

Clock Control Registers

CLKS_CTRL	0xDF23	Controls the "step clock" on CLK_G0
SYNTH_EXT0_CTRL	0xDF24	Controls the PLL settings inCLK_EXT0
SYNTH_EXT1_CTRL	0xDF25	Controls the PLL settings in CLK_EXT1
PENDING_CLKS	0xDF40	Causes clocks G0-G2 to update frequency
G0_INTEGER_B0	0xDFC0	(LSB) Controls frequency of CLK_G0
G0_INTEGER_B1	0xDFC1	(MSB)
G0_FRACTIONAL_B0	0xDFC2	Adjust frequency of GLK_G0
G0_FRACTIONAL_B1	0xDFC3	"
G1_INTEGER_B0	0xDFC4	(LSB) Controls frequency of CLK_G1
G1_INTEGER_B1	0xDFC5	(MSB)
G1_FRACTIONAL_B0	0xDFC6	Adjust frequency of CLK_G1
G1_FRACTIONAL_B1	0xDFC7	"
G2_INTEGER_B0	0xDFC8	(LSB) Controls frequency of CLK_G1
G2_INTEGER_B1	0xDFC9	(MSB)
G2_FRACTIONAL_B0	0xDFCA	Adjust frequency of CLK_G2
G2_FRACTIONAL_B1	0xDFCB	"

Misc Control Registers

PENDING_RST	0xDF4C	Sends a pulse on the user reset (button).

Information Registers

SERIAL_NUM_BYTE0	0xDFF6	Board serial number (ASCII)
SERIAL_NUM_BYTE1	0xDFF7	Board serial number (ASCII)
SERIAL_NUM_BYTE2	0xDFF8	Board serial number (ASCII)
SERIAL_NUM_BYTE3	0xDFF9	Board serial number (ASCII)
MCU_STUFFING1	0xDF27	Bit field indicates which FPGA are installed
TEMP_A	0xDF50	Temperature of FPGA A in units C (binary)
TEMP_B	0xDF51	Temperature of FPGA B
TEMP_Q	0xDFE0	Temperature of FPGA Q
FPGA_A_TYPE	0xDF78	Which type of FPGA is A (encoded)

FPGA_B_TYPE	0xDF79	Which type of FPGA is B (encoded)
CONFIG_ VERSION	0xDFFB	Version of Spartan Firmware
MCU_VERSION	0xDFFC	Version of MCU (“flash”) firmware
BOARD_VERSION_NEW	0xDFFE	Type of board (9200K10PCIE8T) encoded.

4.8.1 Undocumented controls

There are some features that aren’t documented because then we couldn’t change them. If you need a certain feature, email support@dinigroup.com and ask if we are interested in implementing it.

- Turn off auto-increment on USB
- Main Bus error detection
- Main Bus timeout change
- PCI Register read timeout change

4.9 Firmware

A Spartan 3 FPGA and a Cypress micro controller control the configuration circuitry. The programming data for the FPGA is stored on a flash device, and the code for the micro controller is stored on a separate flash device. The instructions for updating the firmware are given in the software section. The flash that stores the Spartan FPGA programming information is made available via a JTAG header, which can be used with the Xilinx program iMPACT. The Dini Group does not recommend doing any sort of development on this FPGA, because if you add custom code, you will not be able to use firmware updates from Dini Group without merging it with your custom code.

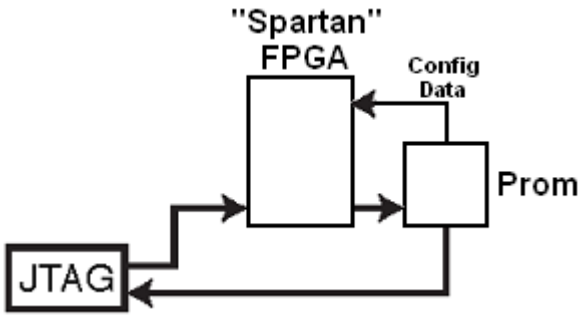


Figure 43 - Spartan "Firmware" JTAG Chain

There is a JTAG chain and header (J6) that is connected to the Spartan and its configuration prom.

Instructions for updating the firmware are in the Controller software section. The Spartan configures from a Xilinx configuration PROM. The microcontroller boots from an IIC EEPROM. It then runs additional code off an external Flash device. The LXT FPGA configures from an external SPI Flash.

5 Clock Network

The board provides a bunch of clocks that go to both FPGAs on GC pins on the FPGA. These clocks are suitable for synchronous communication between the FPGAs. When this manual refers to a "clock input" of an FPGA, it means the "GC" pin described in the Virtex-5 user manual. These pins have the capability of driving a DCM, PLL, or BUFG input with a known (accounted for) delay within the FPGA.

Almost without exception, and clock (or edge sensitive signal) should connect only to a GC pin on the FPGA.

5.1 Global Clocks

All of the "global clock networks" on the DN9200K10PCIE8T are LVDS, point-to-point signals. The arrival times of the clock edges at each FPGA are phase-aligned (length-matched on the PCB) within about 100ps. These clocks are all suitable for synchronous communication among FPGAs.

Since LVDS is a very low voltage-swing differential signal, you cannot receive these signals without using a differential input buffer. Single-ended inputs will not work. An example Verilog implementation of a differential clock input is given below.

```
Wire aclk_ibufds;
IBUFGDS G0CLK_IBUFG (.O(g0clk_ibufg), .I(GCLK0p), .IB(GCLK0n)) ;

always@(g0clk_ibufg) begin
        // Registers
end
```

Either in the UCF or using a synthesis directive, you should set the DIFF_TERM attribute of the IBUFGDS to TRUE. This is recommended because there are no external termination resistors on the DN9200K10PCIE8T.

All global clock networks have a differential test point. The positive side of the differential signal is connected to pin 1 (square) and the negative side is connected to pin 2 (circular).

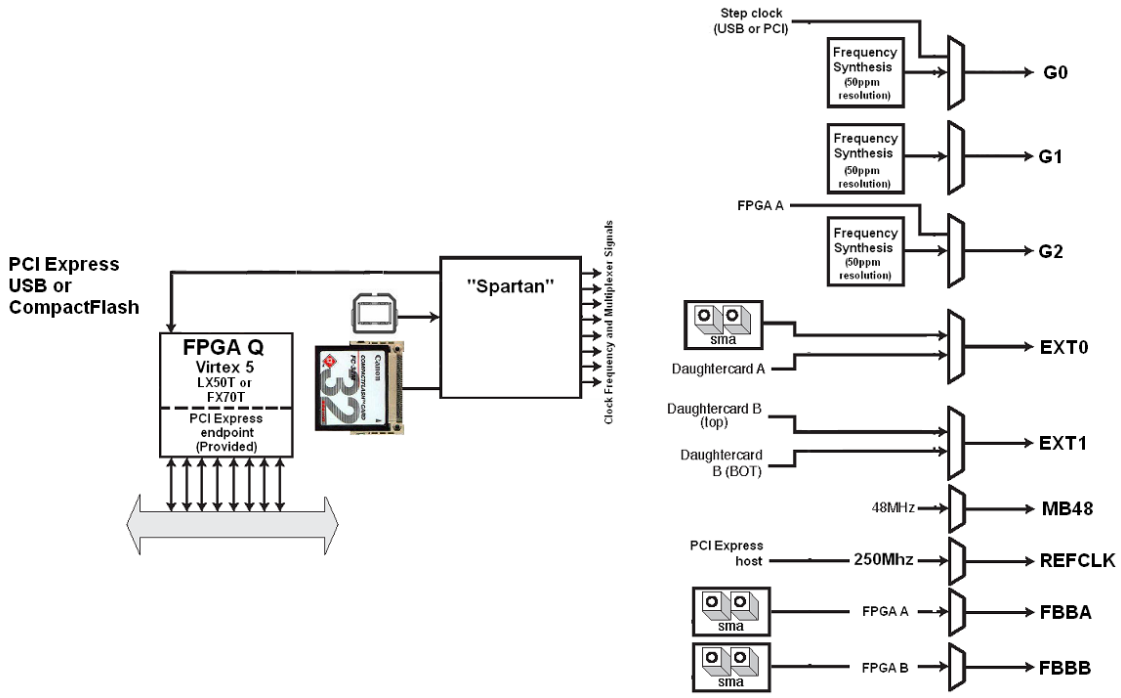


Figure 44 - Clock network block diagram

Each of the nine clock outputs of the clock network is distributed to both FPGAs.

5.1.1 Clock Test points

Each of the "Global clock" networks has a test point. These points are not length-matched with the global clock network, so there may be some phase offset between this point and the FPGA input.

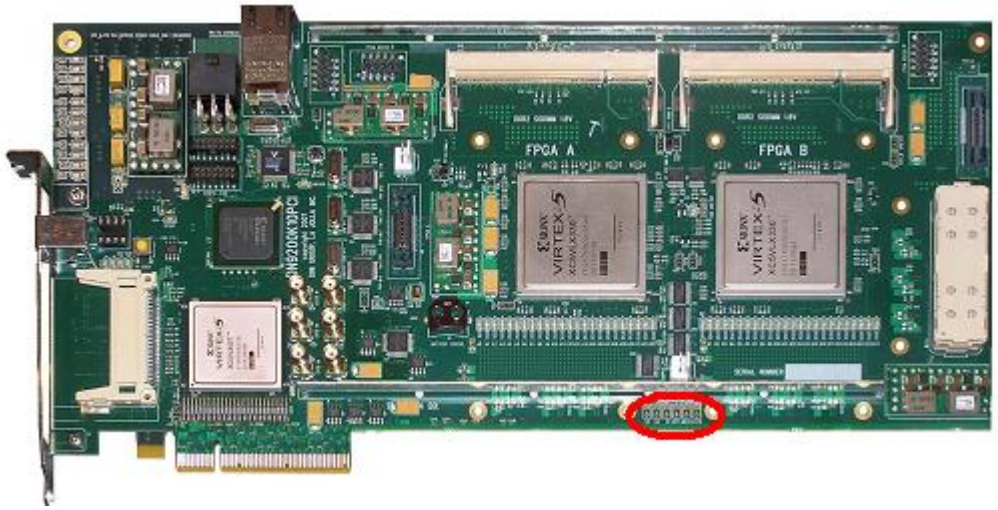


Figure 45 - Clock Test points

All of test points output LVDS signaling. LVDS test points have the "p" signal connected to pin 1 (square) and "n" connected to pin 2 (circular).

A 100Ω resistor connects the P and N side of these clock signals. This is excellent for probing with a high-impedance probe, but not so good for connecting wires. You can remove this resistor if needed.

5.2 G0, G1, G2 Clocks

The G0, G1 and G2 clocks are the primary clock resource for your FPGA design. Each of these clocks can be set to a wide range of frequencies between 0.125 MHz and 550 MHz.

On the schematic, these signals are named
CLK_G*_*p
where * is 0,1 or 2 and * is the name of the FPGA connected to that signal.

Additionally, the reference frequency of each of the can (optionally) come from an alternate source.
G0 can act as a step-clock source controlled via PCI Express or USB.
G1 can be locked to G0
G2 can be controlled from FPGA A

To control the step clock, write to the "configuration register" 0xDF23 using the PCI or USB configuration register interface.

To control the source of each clock, use USB Controller (the "clock sources" option in the Settings menu) or use the SOURCE: command on CompactFlash.

By default, the alternate sources for these clocks are off.

The configuration register that sets the source of the clocks is at location 0xDF16.
bit 0 corresponds to G0, bit 1 corresponds to G1 and bit 2 corresponds to G2. To change the source to the stop clock, write a '1 to the bit location corresponding to the clock network. Then write a '1 to the bit corresponding to the clock network in the "update" register, 0xDF40. Writing to this register will cause a glitch in the clock.

From the compact flash card, source can be set by using the source instruction:

source: G0 2 # sets G0 to step clock 0
source: G1 2 # sets G1 to step clock 1
source: G2 2 # sets G2 to "feedback A"

In USB Controller, from the settings menu, select DN9200K10PCIE8T clock source settings

To control G2 from FPGA A, the FPGA drives a 2.5V clock signal on the CLK_FBA_INT output.

5.2.1 Synthesizer Circuit

The G0, G1, and G2 clock synthesis source is driven by an si5326 clock synthesizer chip. This chip is capable of driving a wide range of output frequencies. The "configuration registers" that control the output frequency are capable of correctly configuring each frequency multiple of 0.125MHz up to 550MHz. If the desired frequency is between one of these steps, or above or below the range, then you will have to use a compact flash card to set the frequency.

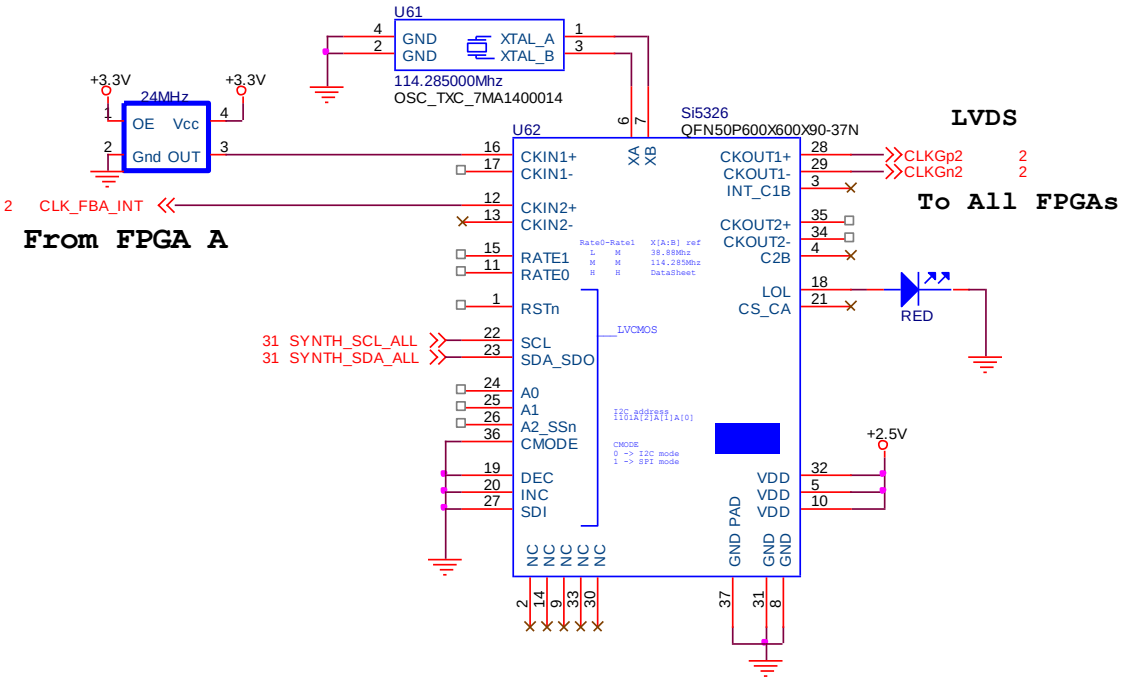


Figure 46 - Clock G network synthesizer circuit

The synthesizer outputs can be set to any frequency within the capability of the synthesizer device. However, the microcontroller cannot calculate the correct settings on the synthesizer because it would require math. In order to obtain an arbitrary frequency setting, you must use the main.txt file on the compact flash card. The main.txt lines required to set clock G1 to a large number of frequencies are given below.

SOURCE: G1	1	7	29393	1599	7	146969	#	0.003000 MHz
SOURCE: G1	1	1	969	23	6	96999	#	0.005000 MHz
SOURCE: G1	1	1	969	23	6	48499	#	0.010000 MHz
SOURCE: G1	1	6	44035	2178	3	44035	#	0.015734 MHz
SOURCE: G1	1	5	22453	999	5	22453	#	0.024000 MHz
SOURCE: G1	1	3	10825	374	3	21651	#	0.032000 MHz
SOURCE: G1	1	7	63915	3478	7	13455	#	0.032768 MHz
SOURCE: G1	1	4	15787	624	4	15787	#	0.038400 MHz
SOURCE: G1	1	7	139971	7618	7	9997	#	0.044100 MHz
SOURCE: G1	1	7	9185	499	7	9185	#	0.048000 MHz
SOURCE: G1	1	1	969	23	6	9699	#	0.050000 MHz
SOURCE: G1	1	3	5773	199	3	11547	#	0.060000 MHz
SOURCE: G1	1	2	10777	319	2	10777	#	0.075000 MHz
SOURCE: G1	1	5	168383	7498	5	7015	#	0.076810 MHz
SOURCE: G1	1	5	5613	249	5	5613	#	0.096000 MHz

SOURCE: G1	1	1	969	23	6	4849	#	0.100000 MHz
SOURCE: G1	1	0	4041	79	4	4041	#	0.150000 MHz
SOURCE: G1	1	3	72667	2516	3	3927	#	0.176400 MHz
SOURCE: G1	1	4	3157	124	4	3157	#	0.192000 MHz
SOURCE: G1	1	7	1377	74	4	2755	#	0.220000 MHz
SOURCE: G1	1	3	13857	479	3	2131	#	0.325000 MHz
SOURCE: G1	1	7	1377	74	4	1377	#	0.440000 MHz
SOURCE: G1	1	3	13857	479	6	1065	#	0.455000 MHz
SOURCE: G1	1	7	1377	74	0	1377	#	0.880000 MHz
SOURCE: G1	1	4	15791	624	3	375	#	1.843199 MHz
SOURCE: G1	1	4	15791	624	3	281	#	2.457600 MHz
SOURCE: G1	1	4	47487	1874	3	211	#	3.276800 MHz
SOURCE: G1	1	5	7909	351	2	225	#	3.579545 MHz
SOURCE: G1	1	4	15791	624	3	187	#	3.686399 MHz
SOURCE: G1	1	7	2303	124	7	107	#	4.096000 MHz
SOURCE: G1	1	6	36307	1790	6	115	#	4.194304 MHz
SOURCE: G1	1	6	49867	2462	0	273	#	4.433617 MHz
SOURCE: G1	1	7	2303	124	7	89	#	4.915200 MHz
SOURCE: G1	1	4	631	24	1	157	#	6.144000 MHz
SOURCE: G1	1	4	15791	624	3	93	#	7.372799 MHz
SOURCE: G1	1	7	2303	124	7	53	#	8.192000 MHz
SOURCE: G1	1	1	2153	52	7	49	#	8.867238 MHz
SOURCE: G1	1	7	2303	124	7	47	#	9.216000 MHz
SOURCE: G1	1	4	15871	624	4	61	#	9.830400 MHz
SOURCE: G1	1	2	507	14	6	47	#	10.160000 MHz
SOURCE: G1	1	3	23221	799	3	67	#	10.245000 MHz
SOURCE: G1	1	7	2303	124	7	39	#	11.059200 MHz
SOURCE: G1	1	5	5613	249	5	47	#	11.228000 MHz
SOURCE: G1	1	3	3611	124	1	85	#	11.289600 MHz
SOURCE: G1	1	7	2303	124	7	35	#	12.288000 MHz
SOURCE: G1	1	3	2549	87	6	33	#	14.318181 MHz
SOURCE: G1	1	7	2303	124	7	29	#	14.745599 MHz
SOURCE: G1	1	4	383	14	6	29	#	16.384000 MHz
SOURCE: G1	1	5	14111	624	5	31	#	16.934400 MHz
SOURCE: G1	1	0	190485	3735	2	45	#	17.734475 MHz
SOURCE: G1	1	0	6085	119	4	33	#	17.900000 MHz
SOURCE: G1	1	7	2303	124	7	23	#	18.432000 MHz
SOURCE: G1	1	4	383	14	4	31	#	19.200000 MHz
SOURCE: G1	1	5	269	11	1	49	#	19.440000 MHz
SOURCE: G1	1	1	31249	767	1	49	#	19.531250 MHz
SOURCE: G1	1	4	15871	624	0	61	#	19.660800 MHz
SOURCE: G1	1	7	2303	124	7	19	#	22.118400 MHz
SOURCE: G1	1	7	2303	124	7	17	#	24.576000 MHz
SOURCE: G1	1	1	3909	95	0	45	#	26.562500 MHz
SOURCE: G1	1	4	383	14	1	29	#	32.768000 MHz
SOURCE: G1	1	7	605	31	1	29	#	33.330000 MHz
SOURCE: G1	1	5	1133	49	5	13	#	38.880000 MHz
SOURCE: G1	1	7	403	19	6	7	#	66.660000 MHz
SOURCE: G1	1	7	6749	363	7	5	#	74.175824 MHz
SOURCE: G1	1	4	383	14	4	7	#	76.800000 MHz
SOURCE: G1	1	5	575	24	4	7	#	77.760000 MHz
SOURCE: G1	1	4	383	14	1	9	#	98.304000 MHz
SOURCE: G1	1	4	383	14	6	3	#	122.880000 MHz
SOURCE: G1	1	5	575	24	6	3	#	124.416000 MHz
SOURCE: G1	1	0	26665	479	6	3	#	133.330000 MHz
SOURCE: G1	1	5	575	24	4	3	#	155.520000 MHz
SOURCE: G1	1	4	9765	374	4	3	#	156.256000 MHz
SOURCE: G1	1	1	509	11	4	3	#	159.375000 MHz
SOURCE: G1	1	7	485	24	4	3	#	160.380000 MHz
SOURCE: G1	1	0	10741	199	4	3	#	161.130000 MHz

SOURCE: G1	1	4	50353	1874	4	3	#	161.132800 MHz
SOURCE: G1	1	3	1173	39	1	5	#	164.360000 MHz
SOURCE: G1	1	0	33325	639	1	5	#	166.630000 MHz
SOURCE: G1	1	0	333333	6399	1	5	#	166.667000 MHz
SOURCE: G1	1	5	92961	3999	1	5	#	167.331600 MHz
SOURCE: G1	1	0	2157	39	1	5	#	172.640000 MHz
SOURCE: G1	1	3	11557	399	3	3	#	173.370000 MHz
SOURCE: G1	1	3	1173	39	3	3	#	176.100000 MHz
SOURCE: G1	1	3	8841	299	3	3	#	176.840000 MHz
SOURCE: G1	1	4	671	24	3	3	#	184.320000 MHz
SOURCE: G1	1	3	6249	191	3	3	#	195.312500 MHz
SOURCE: G1	1	3	2961	99	4	1	#	311.010000 MHz

5.3 Ext Clocks

There are two clock networks on the DN9200K10PCIE8T that are designed to provide clocks from an external frequency reference. EXT0 and EXT1. Each of these clocks is delivered synchronously to both FPGAs and is suitable for synchronous communication among the FPGAs.

EXT0 can be sourced from either the external clock input SMAs connectors or the daughtercard attached to FPGA A (DCA). By default, EXT0 is set to be sourced from the DCA.

EXT1 can be sourced from either DCBB (DaughterCard on fpga B on the Bottom) or DCBT (DaughterCard on fpga B on the Top). By default, the source is DCBB.

The source settings can be made from the USB Controller by selecting menu settings->global clock muxes.

To make the setting from the compact flash card, in the main.txt file, use the MEMORY MAPPED: command to write to the EXT0 register 0xDF27 or the EXT1 register 0xDF28. The register bit map is as follows:

0xDF28[4:0] = S23, S1, S0, PLLSEL, CLKSEL
Write value 0x02 to select the daughtercard
Write value 0x01 to select the FBA clock.

Example: Set EXT0 to use SMA (PLL off):
MEMORY MAPPED: 0xDF27 0x1D

Example: Set EXT1 to use DCBB (PLL off):
MEMORY MAPPED: 0xDF28 0x1C

5.3.1 Daughtercard zero-delay mode

EXT0 and EXT1 can be set to zero-delay mode, where each FPGA is able to receive the clock synchronous to the daughtercard. This feature requires configuring the clock distribution network with the frequency of the clock.

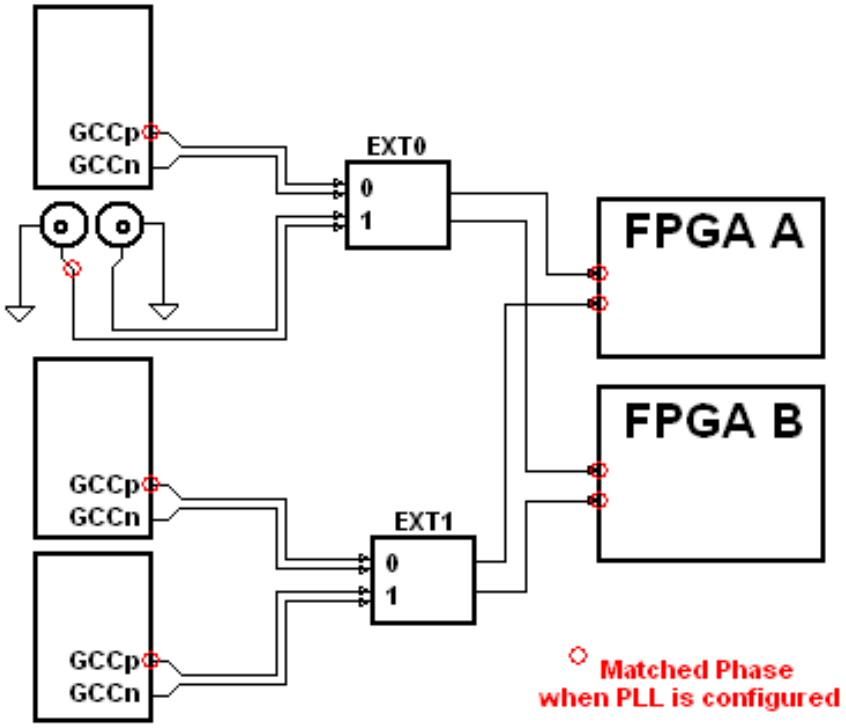


Figure 47 - EXT clock sources diagram

Before you implement read the daughtercard section for more clocking ideas. To set the PLL correctly, use the DCLK command in the main.txt file. For other PLL features such as frequency range and divide/multiply, you must read the PLL data sheet and use the MEMORY MAPPED command in the main.txt file to set the S0 S1 S2 and S3 signals of the PLL.

Note that the phase matching between the FPGAs and connectors is from the FPGA pins to the daughtercard pins, and from the SMA connector to the FPGA pins, therefore, the delay on the daughtercard and on the SMA Cable is not accounted for. The default for the PLL is OFF, so, by default, the phase matching does not occur.

5.3.2 SMA input

The EXT0 clock can be sourced from a pair of SMA inputs, J10, J11. These SMAs connectors can be connected to a differential source or a single-ended source. For single-ended, connect to either the P or N connector. The voltage swing must be between 0.15V and 3.3V.

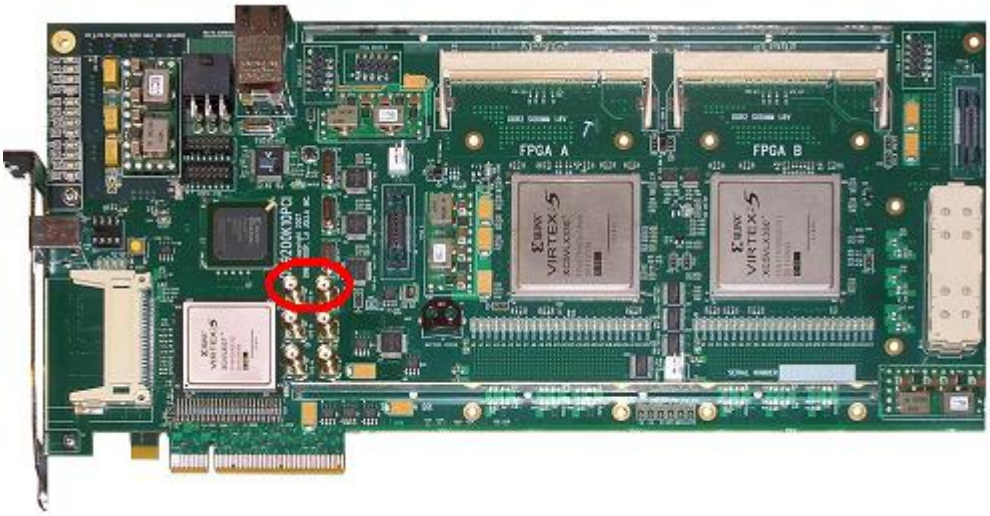


Figure 48 - EXT0 SMA locator

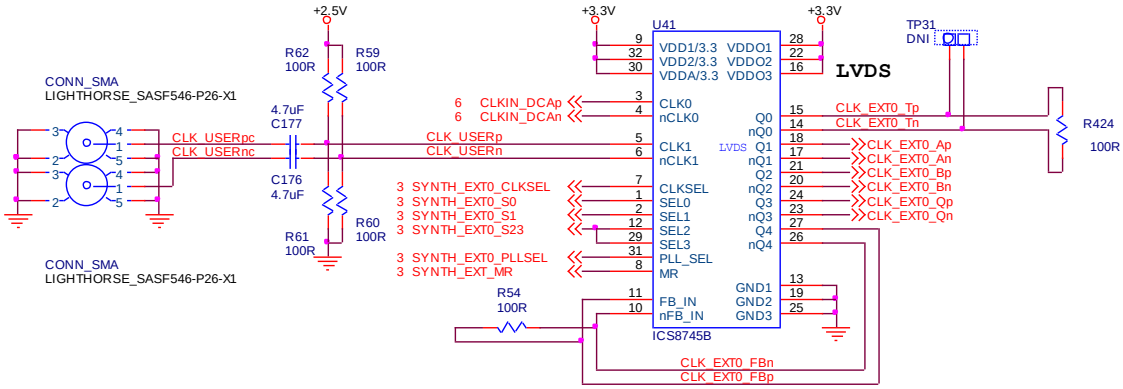


Figure 49 - EXT0 SMA circuit

The inputs are AC-coupled. This limits the *minimum* possible frequency of the clock input to around 50 kHz. If you require an external clock with a frequency lower than this, you should modify the board by removing the 4.7µF resistors shown above and replacing them with 0Ω resistors. The maximum recommended swing on the differential inputs is 3.3V.

5.4 MB Clock

This is a differential clock (must use differential input buffers) that is run at a constant 48 MHz. This clock can be used for whatever you want if you want, but it can also be used for the "MainBus" interface that provides access to USB and PCI Express.

5.5 FBA and FBB clocks

The FPGAs A and B are the source of a global clock network designed to allow an FPGA to generate a frequency for all FPGAs. The name of these signals is

FBA network:
FBA_A (FeedBack from fgpa A to fpga B)
FBA_B

FBB network:
FBB_A (FeedBack from fpga B to fpga A)
FBB_B

FPGA A should drive both FBA signals, and FPGA B should drive both FBB signals. FBA_A is driven out of FPGA A back into FPGA A. This signal can be used as an analogue to FBA_B or it can be used as a feedback.

Similarly, FPGA B drives a signal to itself, FBB_B.

The use model for these clocks requires that FPGA A or B drives an identical clock on both legs of the network output and both FPGAs receive an identical clock on their inputs for use in matching clock networks.

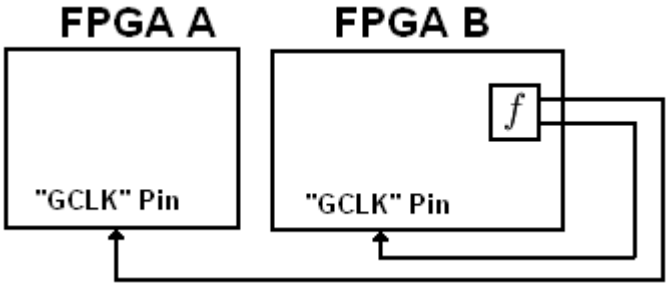


Figure 50 - FBA typical use

You may need to also match this clock's phase with an external phase source. In this case, the feedback signal will need to be used as the feedback to a DCM or PLL. This requirement is common if you have a daughtercard.

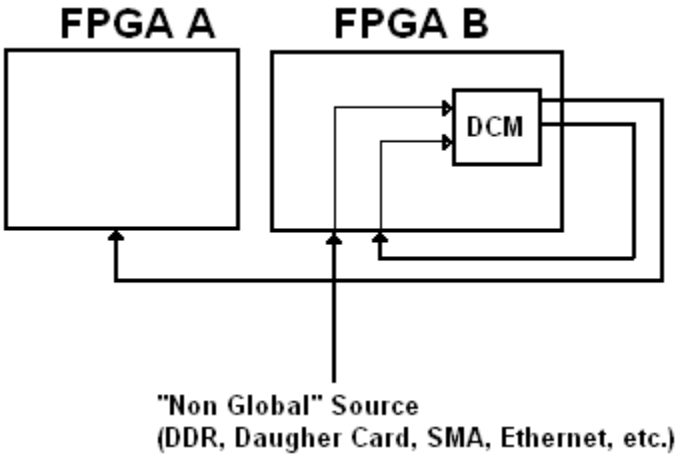


Figure 51 - FBA typical use with synchronization

The additive delays on the feedback network are given below:
FBB 0.86ns
FBA 0.40ns

The FBB network is additionally phase-matched to the daughtercard signals
DCBB0p31 and DCBB0n31
DCBT0p31 and DCBT0n31

The FBA network is additionally phase-matched to the daughtercard signals
DCA0p31 and DCA0n31

This fact can be used to create a low-skew clock to the daughtercards.

5.6 PCI Express REFCLK Network

A clock network driven from the FPGA "Q" is called "REFCLK". When the Dini Group PCI Express endpoint bitfile is loaded into the FPGA "Q", and the board is linked to a motherboard over PCI Express, then this network will be driven with a 250 MHz clock which is equal to 2.5 times the PCI Express REFCLK in frequency.

The network can be used for any other purpose, however, when the FPGA Q is programmed with your own bitfile.

The clock is a differential LVDS signal which should be received on each FPGA with a differential clock input buffer with DIFF_TERM set to TRUE.

When not installed in a PCI Express slot, this clock will be zero MHz (when "full function PCI Express endpoint now with DMA™" core is loaded in FPGA Q).

5.7 Non-Global Clocks

The following sections describe clocks that are not considered "global" because they do not distribute to both FPGAs on the board. These clocks may be used for specific interfaces and details on the clocking required for those interfaces are found in a different section in the hardware chapter.

5.7.1 Clock TP

Each FPGA is connected to a two-pinned test point. This test point can be used to input a differential clock from off-board. Each of these test points has a 100Ω jumper installed shorting the negative and positive signals. To input or output differentially, you must remove this resistor.

The net name on the schematic and in the provided UCF for this signal is
CLK_DIMMB_DQS3p/n and
CLK_DIMMA_DQS3p/n

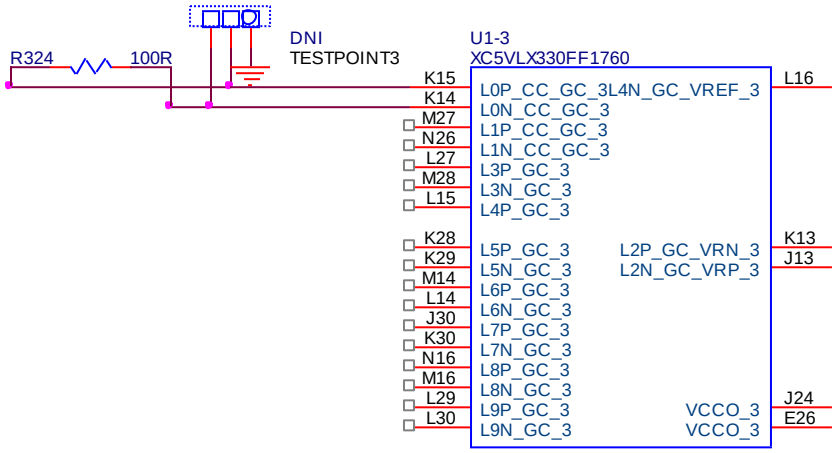


Figure 52 - Clock Testpoint circuit

The schematic clipping above shows FPGA B's test point, but all FPGAs use the same pinout. A list of all test points on the board can be found in the test points section.

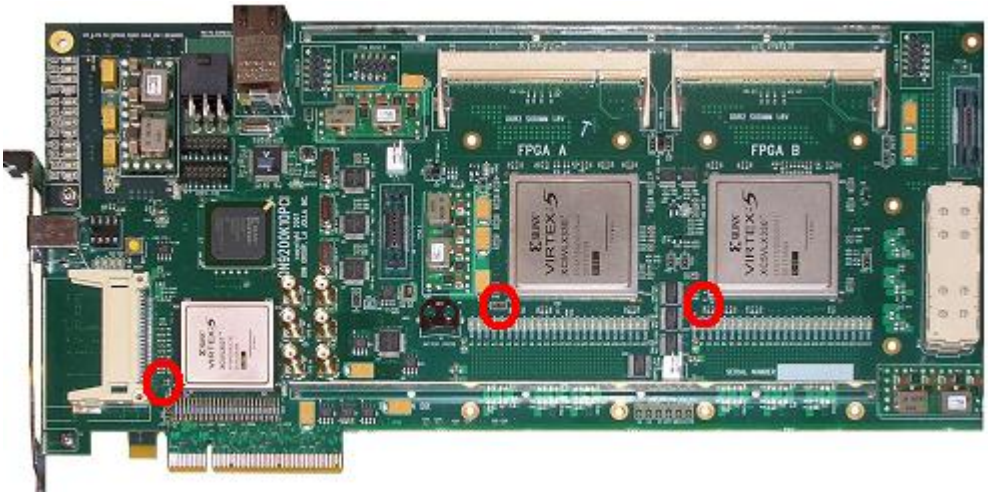


Figure 53 - Clock Test point locator

This signal can also be used as an external feedback path for a DCM. Drive a single-ended clock out the "N" side and receive it on the "P" side. The additive external delay for this feedback path is 1.6ns. The maximum frequency for the feedback path is 250 MHz.

5.7.2 Ethernet Clock

The VSC8601 Ethernet PHY device outputs a 125 MHz clock. The signals in the schematic are CLK125_ETHA. This signal is LVCMOS25, single-ended signals. The frequency is fixed.

This clock input can be used as a general-purpose 125 MHz source.

Details about appropriate clock methodology for the Ethernet interface is in the Ethernet section.

5.7.3 DDR2 Clocks

The CK signals in the DDR2 interface are described in the DDR2 interface section.

OUTPUTS

CLK_DIMMA_CK2p	E39
CLK_DIMMA_CK2n	E40
CLK_DIMMB_CK2p	AC33
CLK_DIMMB_CK2n	AD32

INPUTS

CLK_DIMMA_CK2p	AM13
CLK_DIMMA_CK2n	AN14
CLK_DIMMB_CK2p	AM13
CLK_DIMMB_CK2n	AN14

Note that on the netlist, these signals connect to the FPGA twice: once on the DDR2 interface bank (1.8V), and once on the global clock input bank (2.5V). The 2.5V, clock bank connections should be used as inputs, and the 1.8V bank signals should be configured as outputs. For input signals, use the LVDSEXT standard with the DIFF_TERM attribute set to TRUE.

If the DIMM interfaces are not used, these can be used as external feedback traces. The external delays are given here:

CLK_DIMMA_CK2	0.65ns
CLK_DIMMB_CK2	0.63ns

5.7.4 SMA Clock B and E

All FPGAs have a pair of SMA connector connected directly to global clock inputs. The bank connected to these signals is a +2.5V bank. Allowed input standards are LVCMOS25, SSTL25, LVDS, DIFF_SSTL18.

FPGA A pins AM28, AN28
FPGA B pins AK28, AK27

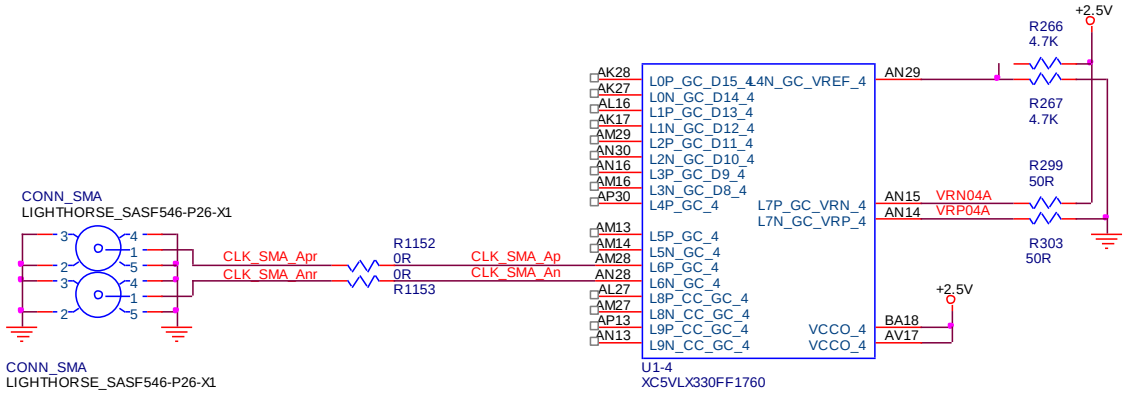


Figure 54 - SMA circuit

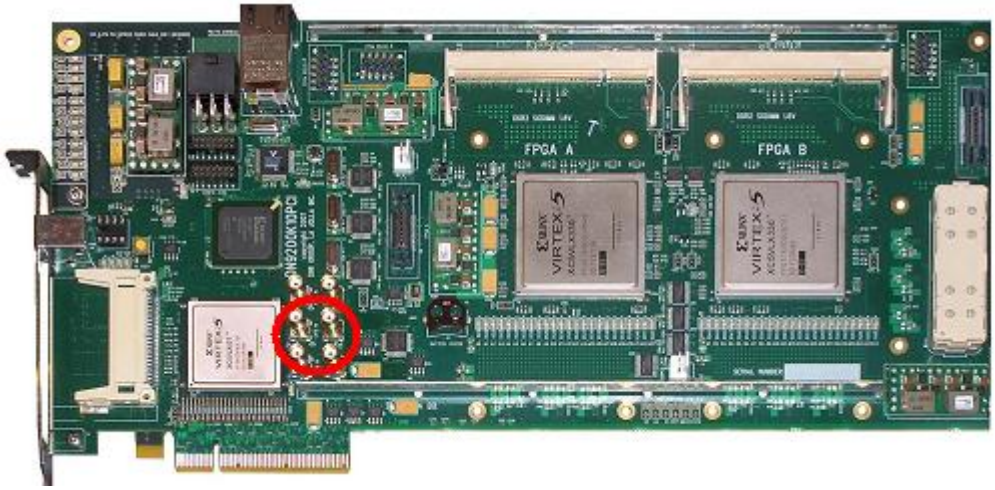


Figure 55 - SMA locator

These connections are DC-coupled, meaning the user must ensure that the levels received on this input are within the limits of the Virtex-5 device to prevent damage to the part.

This pair of SMA connectors can also be used as outputs, as single-ended inputs or for non-clock signals.

DCI is enabled on these inputs. You can use SSTL2_II_DCI as end-terminated inputs.

5.8 Clock Use notes

The following sections give hints for successful clock network design.

5.8.1 Achieving Zero clock-to-out

Many high-speed chips are designed to have a zero hold time requirement on their inputs. This convention is convenient because it means that the optimal output timing is always where a clock edge that it aligned perfectly with the data. In the FPGA, there are two easy ways to achieve this.

1) Output the clock for the external interface from a DDR flip-flop, using the same clock as the output data.

2) Use an external feedback with zero additive phase.

5.8.2 Forwarding Clocks FPGA-to-FPGA

Creating a frequency in one FPGA and sending it to other FPGAs is very common (34.2%). Often a clock needs to be dynamically selectable between two sources, or be turned on and off. Or maybe you need to do a multiplication or division of a clock, or you want your entire system to be clocked off a single frequency, provided by an interface only available to one FPGA. In this case, you need a low-skew way to forward clocks from FPGA-to-FPGA.

First, please consider using the FBA and FBB clock networks. This is exactly what these networks are intended for. There are other available methods, however. I've listed some of them here (in order of how good I think they are), but I'm sure there are others.

1) Use the FBA and FBB networks
2) Use the FBA_INT signal to control the frequency of clock G2
3) Drive the clock onto a daughtercard and feed it back to the EXT0 or EXT1 network. (We can provide a loopback daughtercard if you want).
4) Use one of the global clock networks as a phase source, and over an FPGA interconnect signal send up to 16 synchronous clock enables. Use the BUFGMUX macro in your FPGA to gate the clock. The effective clock periods for the resulting 16 clocks will vary from cycle-to-cycle, however each frequency can be independent.
5) Drive a clock signal out the FPGA to the SMA connector and feed it through a cable back to the EXT0 SMA input.
6) Drive the clock signal on standard IO pins, and use a DCM in the receiving FPGA to dynamically align its clock to the input. Use the DCM's output as a clock and sample the forwarded clock in a flip flop. Then adjust the phase of the DCM's output back and forth so that the logic level on the flip-flop bang-bangs from a 0 to a 1 and so forth.

The following methods are incorrect, but common. Note that if you use one of these methods, it will only work as if there is plenty of time before your project deadline. When the deadline approaches, it will stop working correctly.

5.8.2.1 Always Use GCLK Pin

Customer Ophelia Payne, who has been trying to get a job at Google for 4 years, has routed a signal to a "non GCLK" pin of FPGA A. She uses the signal AB03p13 as a clock from B to A.

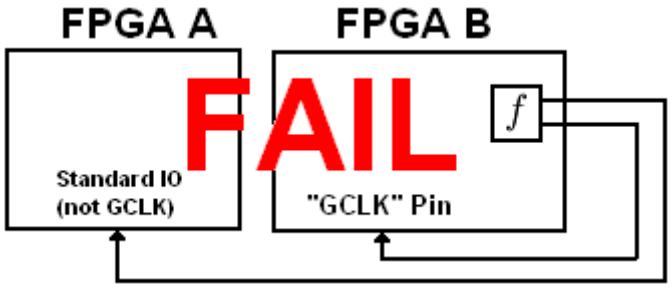


Figure 56 - Not using GCLK pins

Unfortunately, there is a long (13ns) skew from the arrival of the clock to the flip-flops of FPGA A and FPGA B. What's worse, a DCM could be used to account for the delay because the routing between the pins and the DCMs is not in the feedback path! Furthermore, there are degradations in performance of the clock like low maximum frequency, duty cycle distortion, jitter, glitches, low birth weight, poor precision from timing analysis, and inconsistent skew from one place-and-route to another. Ophelia should use a GCLK pin on FPGA A. She should use the FBB* clock network instead.

5.8.2.2 Always use a low-skew network

Mel Loewe, a 12-year ASIC design veteran, has synthesized a frequency in FPGA B. He uses this frequency in FPGA B for his IO outputs, and also drives the clock out to FPGA A, using the FBB network that I told him to use. In FPGA A, this clock comes in on a GCLK pin and is used to clock the inputs of FPGA A.

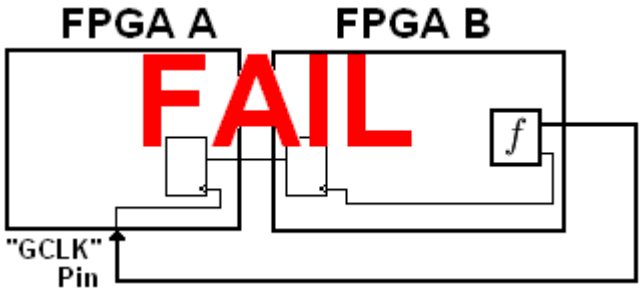


Figure 57 - Not using an external feedback

Oops! Mel has some hold time violations on FPGA B because the external delay of the clock from FPGA B to FPGA A is not mirrored in the clock scheme of FPGA A. Mel should drive the other leg of the FBB* network "FBB_B" from FPGA B to FPGA B, so that both A and B have an external clock trace in the delay path.

5.8.2.3 Synthesized Frequencies

Anita Mann, janitor who found a DN9200K10PCIE8T discarded in a waste bin, has two domains, a 48 MHz core and a related 24 MHz IO. She says, "gee wiz, I'll just divide down that clock in the FPGA!" She knows that the DCM is guaranteed to have zero skew between inputs and divided outputs, and therefore zero skew between the two FPGAs.

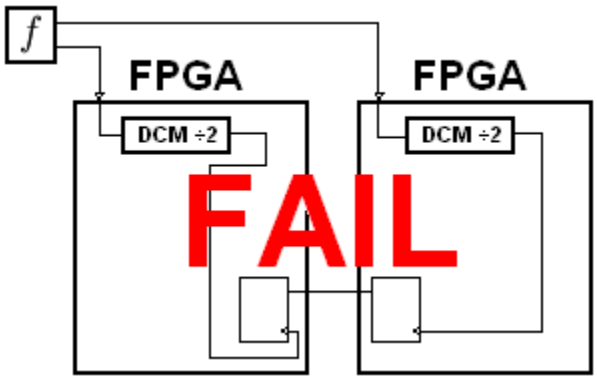


Figure 58 - Two divide DCMs

Wait a second, Ann. There are two valid output phases from a divide-by-two operation, each 180° apart from the other. One of the FPGA's clock might have opposite polarity as the other. Ms. Mann could have distributed a 24 MHz clock and multiplied by 2. Or she could send some sort of synchronization signal across the FPGAs. Finally, she could synthesize the divide-by-two clock in one FPGA, then distribute this clock on a network, using one of the methods described in "Forwarding clocks FPGA-to-FPGA".

5.8.2.4 Use an ODDR for clock outputs

Justin Casey Howells III, an untrustworthy vegan, needs to drive a clock out to an external device (or another FPGA). So he looks in his *A Verilog Pocket Reference* that his wife got him for Kwanza. It says that the proper syntax is

```
assign clkout = f;
```

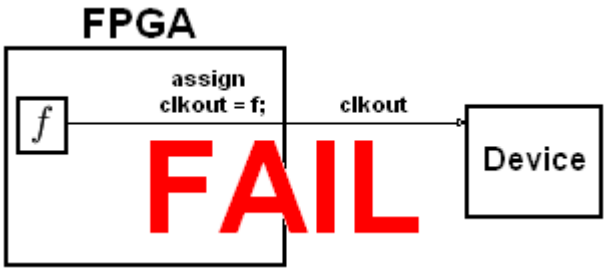


Figure 59 - Outputting a clock with an assign statement

This seems to work fine, except when it failed constantly. The problem here is that the FPGA is incapable of routing a clock signal with low-skew to the output pad. In order to get consistent timing on this signal, Justin should use an ODDR like this:

```
ODDR justins_oddr(.C(f), .I(1'b1), .IB(1'b0), .O(clkout));
```

5.8.2.5 Cascading DCMs

Mickey, a giant talking mouse, decided he needs to synthesize a frequency in one FPGA. Mickey (who is called "Mick" by his friends) knows all about clock phases, so he uses a DCM in the receive FPGAs to dynamically center the clock optimally in the data valid window. Also, he is sure to reset the DCMs, like the Virtex-5 User Guide requires.

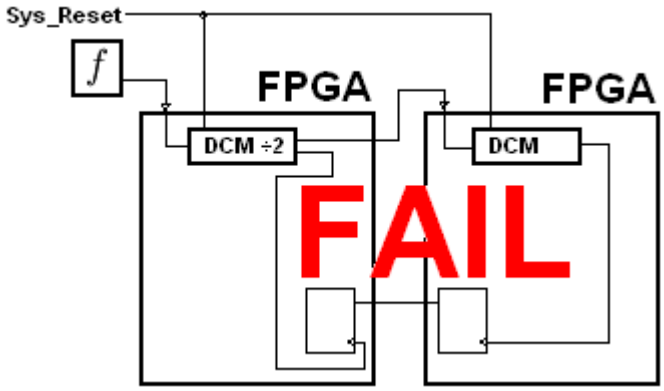


Figure 60 - Cascading DCMs

Mick forgot that before DCM #1 gets it's reset, it isn't outputting a clock, and so the clock input to DCM #2 isn't stable until after reset is released. Oops! Mick should either put a timer on the reset of DCM #2, or else route the LOCKED signal from DCM #1 to the RESET #1 port of DCM #2.

5.8.2.6 DCM Reset Timing

Anna Graham, a tenured professor who couldn't care less about her "research", connects SYS_RESET to her DCM and her logic, like she learned in ASIC camp.

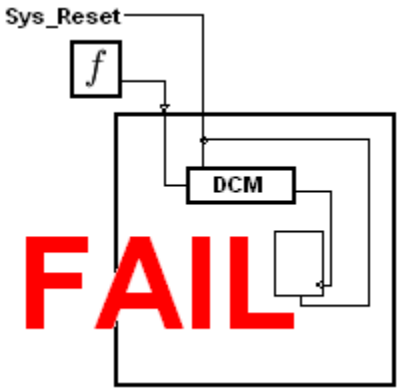


Figure 61 - DCM on same reset as logic

The problem here is that the DCM doesn't output a stable clock until 50us after it receives reset. Now all the flip-flops in her design have to survive 50us of complete pandemonium.

6 Test points

This section lists all of the test points on the DN9200K10PCIE8T. A more detailed description may be found in the section about the system that the test point is part of, but all test points are listed here for reference.

Part Reference	Net name	Purpose
Ground Points		
MP1,MP2	GND	Ground rails good for probe clips
M2,M1,Y2	GND	Ground holes good for mounting board
Power Access Pointes		
TP15	+12V	+12V power from power connector
TP33	+1.0V_A	1V nominal for FPGA A internal power (1.05V actual)
TP40	+1.0V_B	1V nominal for FPGA B internal power (1.05V actual)
TP1	+2.5V	+2.5V for FPGA IO
TP3	+3.3V	+3.3V for configuration circuit
TP28	+5.0V	+5V for daughtercards
TP13	+VDIMM_B	Voltage for DIMM connected to FPGA B (1.5V – 3.3V)
TP16	+VDIMM_A	Voltage for DIMM connected to FPGA A (1.5V – 3.3V)
TP8	+0.9V_B	Half of VDIMM_B (for termination)
TP20	+0.9V_A	Half of VDIMM_A (for termination)
TP32	+1.2_S	+1.2V for Spartan internal
TP41	+3.3_MGT	+3.3V for PCI Express clock synthesizers

TP45	+2.5_MGT	+2.5V for PCI Express clock synthesizers
TP34	+VBATT_TP	Input 1V-3V for Encryption battery

TP27	SYS_RST#	Hardware-generated reset (for power-on)
TP29	BUTTON_S#r	User button ("FPGA Reset")
TP30	ADC_A0p/n	System monitors analogue input for FPGA A
TP31	ADC_B0p/n	System monitors analogue input for FPGA B
TP38	ADC_Q0p/n	System monitors analogue input for FPGA Q
TP35	CLK_DIMMA_DQSp/n3	Connects to "GCLK" pins of FPGA A
TP36	CLK_DIMMB_DQSp/n3	Connects to "GCLK" pins of FPGA B
TP37	CLK_TP_Qp/n	Connects to "GCLK" pins of FPG Q
TP39	CLK_GTP_118p/n	Connects to GTP "refclk" pins of FPGA Q

GLOBAL CLOCK TESTPOINTS

TP47	CLK_G0_Tp/n	these test points are suitable for checking the frequency and stability of the global clock networks
TP48	CLK_G1_Tp/n	
TP43	CLK_G2_Tp/n	
TP49	CLK_MB48p/n	
TP42	CLK_REF_Tp/n	
TP46	CLK_EXT0_Tp/n	
TP44	CLK_EXT1_Tp/n	

Voltage Measurement Test points

TP2	+1.0V_A	these test points are intended for measuring the board voltages; They are located conveniently along the left edge of the board next to LEDs. They are connected to the power supplies with thin wires, so you should not try to draw more than 100mA from these points
TP4	+1.0V_B	
TP7	+2.5V	
TP9	+3.3V	
TP10	+5.0V	
TP5	+VDIMM_A	
TP6	+VDIMM_B	
TP11	+MGT_AVCC	
TP12	+MGT_AVTT	
TP14	+MGT_AVCCPLL	
TP17	+VIO_DCA0	
TP18	+VIO_DCA1	
TP19	+VIO_DCA2	
TP21	+VIO_DCBB0	
TP22	+VIO_DCBB1	
TP23	+VIO_DCBB2	
TP24	+VIO_DCBT0	
TP25	+VIO_DCBT1	
TP26	+VIO_DCBT2	

DIMM signal test points

TP50	DIMMA_CAS#	These signals are under the DIMMs on the

TP51	DIMMA_WE#	back side of the board. They are intended for probing the DDR2 signals for debugging user logic
TP52	DIMMA_DQSp0	
TP53	DIMMA_DQ00	
TP58	CLK_DIMMA_CK0p/n	
TP60	DIMMA_RAS#	
TP54	DIMMB_CAS#	
TP55	DIMMB_WE#	
TP56	DIMMB_DQSp0	
TP57	DIMMB_DQ00	
TP59	CLK_DIMMB_CK0p/n	
TP61	DIMMB_RAS#	

7 USB interface

The DN9200K10PCIE8T allows the user FPGA to communicate to a host PC over USB. The configuration circuitry allows this by bridging USB to the Main Bus interface. For most users, implementing USB communication will be as simple as making a Main Bus controller. In the reference design, there is an example Main Bus controller. See the Main Bus section of this chapter for more information on the Main Bus.

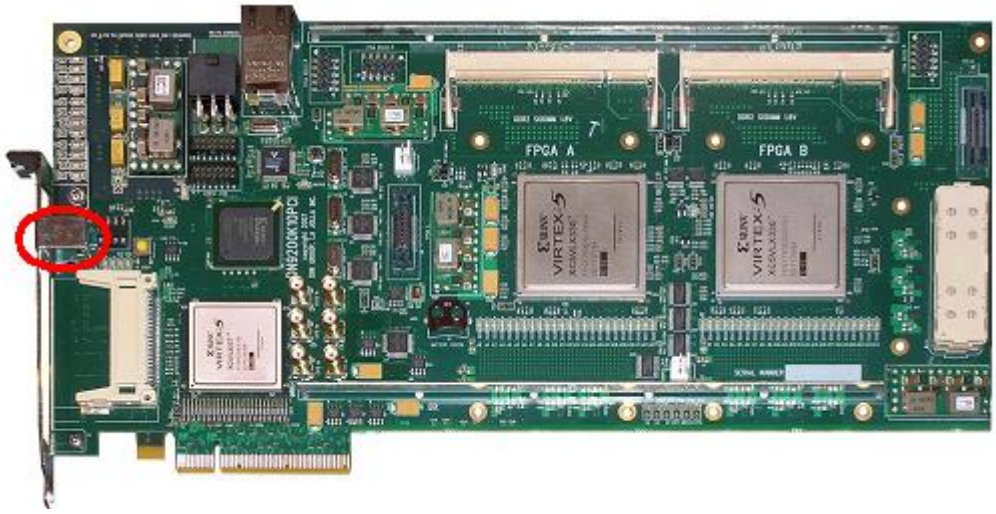


Figure 62 - USB locator

USB on the DN9200K10PCIE8T also allows control of the configuration circuitry from a host PC. This includes configuring FPGAs, setting clock frequencies and others

This section will describe the software interface required to communicate to the DN9200K10PCIE8T. In addition to reading this section, you may chose to modify the provided software (USB Controller and AETest_usb). The source code for these programs is on the user CD. These programs collectively implement all of the available controls on the DN9200K10PCIE8T.

7.1 Vendor Requests

Most of the "control" functions available over USB are accomplished using a "vendor request". Programming a USB vendor request is out of the scope of this document, but you can copy the code provided in the USB Controller program.

The following table describes the USB interface presented to the host by the MCU micro controller.

Vendor Request Name	Code	Purpose
VR_CONFIG	0xaf	Causes FPGAs to configure from CF card
VR_CHECK_FPGA_CONFIG	0xb5	Read the "DONE" status of the FPGA
VR_MEM_MAPPED	0xbe	Write to a "configuration register"
VR_CLEAR_FPGA	0x90	Clear ("PROGn") an FPGA
VR_SET_EP6TC	0xbb	Set the size of Bulk Transfer reads (required)
VR_SET_EP2TC	0xba	Set the size of Bulk Transfer reads (required)
VR_SETUP_CONFIG	0xb7	Put the USB Endpoint into configure mode
VR_END_CONFIG	0xbd	End configuration mode (required)
VR_ENABLE_MSD	0xC0	Put the USB endpoint in card reader mode
VR_DISABLE_MSD	0xC1	Finish card reader mode
VR_DEFAULT_ENABLE_MSD	0xC2	Put the USB endpoint in card reader mode
VR_DEFAULT_DISABLE_MSD	0xC3	Finish card reader mode (permanent)
VR_FLASH_VERSION	0xb2	Read "flash" firmware version
VR_SM_CD	0xb8	
VR_BOARD_VERSION	0xb9	Read the type of board (DN9200K10PCIE8T)
FLASH_VERSION_ADDR	0x08	Read "flash" firmware version again

Each vendor request has a direction, request type, request, and value, size and buffer pointer fields. The request type is always TYPE_VENDOR. The request field is the ID listed in the table above. The value and data in the buffer pointer fields are vendor-request specific. The size field is the number of bytes in the buffer. The details of how to implement a vendor request are outside the scope of this manual.

7.1.1 VR_CLEAR_FPGA

This vendor request clears an FPGA.

Direction is OUT. Size is 0. Value represents which FPGA should be cleared. 0 is FPGA A. 1 is FPGA B… and so on.

7.1.2 VR_SETUP_CONFIG

This vendor request must be called before sending configuration data to an FPGA. It tells the DN9200K10PCIE8T which FPGA should receive the next configuration stream sent over USB. It also clears that FPGA of its current configuration.

Direction is OUT. Size is 1. In the buffer is a number representing which FPGA should be selected. 0 is FPGA A, 1 is FPGA B, 2 is FPGA C… and so on.

7.1.3 VR_END_CONFIG

This vendor request de-selects and FPGA (so that configuration data sent will go to no FPGA) and checks the configuration status of an FPGA.

7.1.4 VR_SET_EP6TC (Read buffer size)

The SetReadBufferSize vendor request must be used before any "bulk read" bulk transfer. This sets the size (in bytes) of the data that will be requested by the bulk transfer. If this vendor request is not sent before the bulk read, the behavior is undefined.

The direction is OUT. The size is 0. The value is the number of bytes required for the next bulk transfer.

7.1.5 VR_MEM_MAPPED (Configuration Registers)

This Vendor request allows access to the "Configuration Registers" on the board. These are primarily required for configuring clocks. A full list of these is given in the "Configuration Section"

To write to a configuration register, use the VR_MEMORY_MAPPED vendor request.

The direction is OUT. The "value" field is the address you wish to write to (example 0xDF39, the disable Main Bus register). The size field should be 1. The buffer should contain a single byte containing the byte to be written to the Configuration Register. All configuration registers are one byte.

7.2 Main Bus Accesses

The only way to get user data to and from the FPGA is to use the Main Bus interface. To implement a MainBus slave on your FPGA, see the Main Bus section in the Hardware chapter.

To request a Main Bus interface write transaction, the USB Controller program sends a USB bulk write to EP2 (endpoint 2). The first byte contains an op code, (0x00 or 0x01), determining whether the next 4 bytes contain an address or a datum. If this byte is a 0x00, the next 4 bytes in the bulk transfer are stored into an address register. All data transferred to and from the main bus is LSB first.

Example: Set the current MainBus address to 0x18000000.
Send a Bulk Transfer OUT request to endpoint 2, of length 5 bytes: 0x00, 0x00, 0x00, 0x00, 0x18.

Example: Write the data 0xFF00FFAA to the current MainBus address.
Send a Bulk Transfer OUT request to endpoint 2, of length 5 bytes: 0x01, 0xAA, 0xFF, 0x00, 0xFF.

If a sequence is sent that does not start with a known op code, or the data afterwards is of an unexpected length, MainBus and/or USB will hang.

After each data word is sent, the current address on MainBus automatically increments to the next address. (This behavior can be disabled).

To request a main bus read operation, the USB Controller sends a USB bulk write to EP2 to set the address register, as described in the above paragraph. Then, the USB Controller sends a bulk read to EP6 (endpoint 6), with the USB bulk request SIZE field set to the number of bytes requested. The number of bytes requested must be divisible by 4. After the bulk read is complete, the address register is incremented by SIZE ÷ 4.

Before starting a USB read using a bulk transfer, you must tell the DN9200K10PCIE8T how many bytes are going to be read by using the VR_SET_EP6TC (0xBB) vendor request described in the *Vendor Requests* section.

Example: Read one MainBus DWORD from address 0x18000004.
Send a Bulk Transfer OUT request to endpoint 2, of length 5 bytes: 0x00, 0x04, 0x00, 0x00, 0x18.
Send a Vendor Request of type VR_SET_EP6TC with value of 4.
Send a Bulk Transfer IN request to endpoint 2 of size 4.

Notice that using the above methods; the write bandwidth is limited by the overhead of interleaving op-codes and data. A method of writing to Main Bus with a smaller overhead is the op code 0x03. Using this op code, the 4 bytes following the op code give a number of DWORDs that will follow, which are all data that should be written to consecutive MainBus addresses. This data must be of a length divisible by 4.

Example: Write the data pattern 0xFFFFFFFF, 0x00000000 to MainBus address 0x18220016.
Send a Bulk Transfer OUT request to endpoint 2 containing the data sequence:
0x00,0x16,0x00,0x22,0x18,0x03,0xFF,0xFF,0xFF,0xFF,0x00,0x00,0x00,0x00

7.2.1 Note about Endpoint Terminology

In USB an endpoint is either read or write. It is either for Vendor Requests, or for Bulk Transfers

2 – Host-to-board (Main Bus, FPGA Configuration, Prom JTAG)
4 – Host-to-board (Mass Storage mode)

6 – Board-to-host (Main Bus, Readback, Prom JTAG)
8 – Board-to-host (Mass Storage mode)

In the Windows USB model, there are "Pipes" that can be used for bulk transfers. Which pipe connects to which endpoint is determined dynamically by the Windows driver subsystem. Since some of the endpoints on the Dini Board can be enabled or disabled, the correct windows "pipe" to use for a given function can change. Therefore, the user should iterate through the available pipes and check their endpoint numbers.

In the Linux USB model (either usbdevfs or usblib), endpoints are colloquially numbered by the endpoint number byte in a USB packet, where the MSB describes the direction of the endpoint. Therefore in Linux code, the endpoints may be numbered 0x02, 0x84, 0x06 and 0x88.

Endpoint 0 is a control (vendor request only) endpoint and the driver will automatically specify endpoint 0 when the vendor request function is called. Users can pretend it does not exist.

7.2.2 Performance

Main Bus over USB runs at a maximum speed of 80 Mbs in either direction. This number assumes that the FPGA operates the Main Bus interface with zero wait cycles. If the FPGA design has more wait cycles, this speeds decreases. The approximate speed of Main Bus over USB is given below as a function of Main Bus wait states.

0 cycles	80Mbs read
1 cycle	76Mbs read
5 cycles	64Mbs read
30 cycles	32Mbs read
100 cycles	13Mbs read
250 cycles	6Mbs read

Also, each USB operation requires about 0.5 ms of latency. So for small Bulk Transfers, bandwidth will be limited. The code provided with the board is not as efficient is possible. For each Main Bus read, for example, it might write a Vendor Request to enable USB, one Bulk Transfer to set the Main Bus address, one Vendor Request to set the endpoint read size to 4, and one Bulk Transfer to read the data.

Here are ways performance can be improved:
- Keep track of the current read size in the host software.
- Keep track of the current MainBus address in the host software.
- Make MainBus registers consecutive so that reads and writes don't require changing the address
- Always pack consecutive writes and address changes into one Bulk Transfer
- Always keep reads the same size

7.3 FPGA Configuration Mode

Instructions for programming FPGAs over USB can be found under "Configuration Section"

7.4 Mass Storage Device Mode

When a certain vendor request is made, the MainBus endpoint is replaced by the CompactFlash card slot on the board, which will appear to the computer as a Mass Storage Device. From Windows, or another operating system, you can read and write files to the CompactFlash card.

While you are in this mode, Main Bus cannot be used over USB.

7.5 Firmware Update Mode

When a certain vendor request is made, the Main Bus endpoint is put in Firmware Update mode. The interface in this mode is not described here. It's purpose is to allow firmware updates for customers that do not have a JTAG cable. However you probably do have this cable because it's very useful.

7.5.1 Activity LED

A yellow LED located next to the USB connector flickers when there is USB activity.

7.6 Hardware

The USB hardware implementation is not documented, but I'm sure you can figure it out from the schematic.
- USB is Hot-Swappable
- DN9200K10PCIE8T does not draw power from USB

7.6.1.1 Cypress CY7C68013A

The Physical USB interface is provided by a Microcontroller. You do not need to know anything about it. The code is provided if you care.

The source code for the MCU firmware ("Flash") is provided in
D:\Config_Section_Code\MCU
as a Keil Studios MicroVision 2.11 project file.

7.7 Troubleshooting

If you cannot get USB to communicate with your design over Main Bus, please try using the USB Controller software with your design, and using the Dini Group reference design with your software. This will help determine whether the software or the hardware is causing the error.

If USB appears to not work at all, try connecting to a Windows computer, and checking if the device shows up in the Device Manager. If so, then the Hardware is working correctly and there is a driver or software problem. If not, there is a hardware problem. (Board stuck in reset? Bad firmware update?)

7.7.1 USB Controller Freezes

The Vendor requests on the DN9200K10PCIE8T are blocking. Only one can be completed at a time. This includes vendor requests that take a very long time like "Configure from

CompactFlash" (10 seconds). During this time USB Controller, a single-threaded application, freezes when any Vendor Request is issued. (All the time). If a process fails, USB Controller will hang forever. You can unplug USB or turn off the board, and USB Controller will work again.

The normal way to avoid problems like this is to create a separate hardware-IO thread.

8 FPGA Q Resources

8.1 FPGA A Interconnect

The interconnect between FPGA A and FPGA Q is single-ended only. However it is also completely length-matched, with an additive delay of 1.12ns. The clocks PCIE_PCLKA and PCIE_PCLKQ are also matched to this length, making the interface perfect for a source-synchronous interface with no per-bit alignment required. The maximum frequency achievable using this method is about 300 MHz

8.2 Unusable IO

FPGA Q has some IO pins that are connected directly to ground. These pins are AA5, AB5, AF4, AF3, A3, B4, B5, D5, E5. It is recommended that you drive these pins with a constant low value, and assign a high drive-strength driver to the IO type. These pins are intended to help shield the sensitive RocketIO power supply pins from IO switching noise.

8.3 RocketIO ("MGT", "GTP", "GTX")

All 8 of the available serial channels on this board are used for PCI Express. They cannot be used for anything else unless you plug into some sort of adapter card. If you really want, we can provide this for you. It might look like this:

Figure 63 - PCI SIG Compliance Base Board

You may notice that the FX70T actually has 12 GTX and not 8 like I say. Trust me, these cannot be used because in the small package, the extra 4 GTX channels do not connect to pins on the package.

8.4 SPI Flash

8.5 LEDs

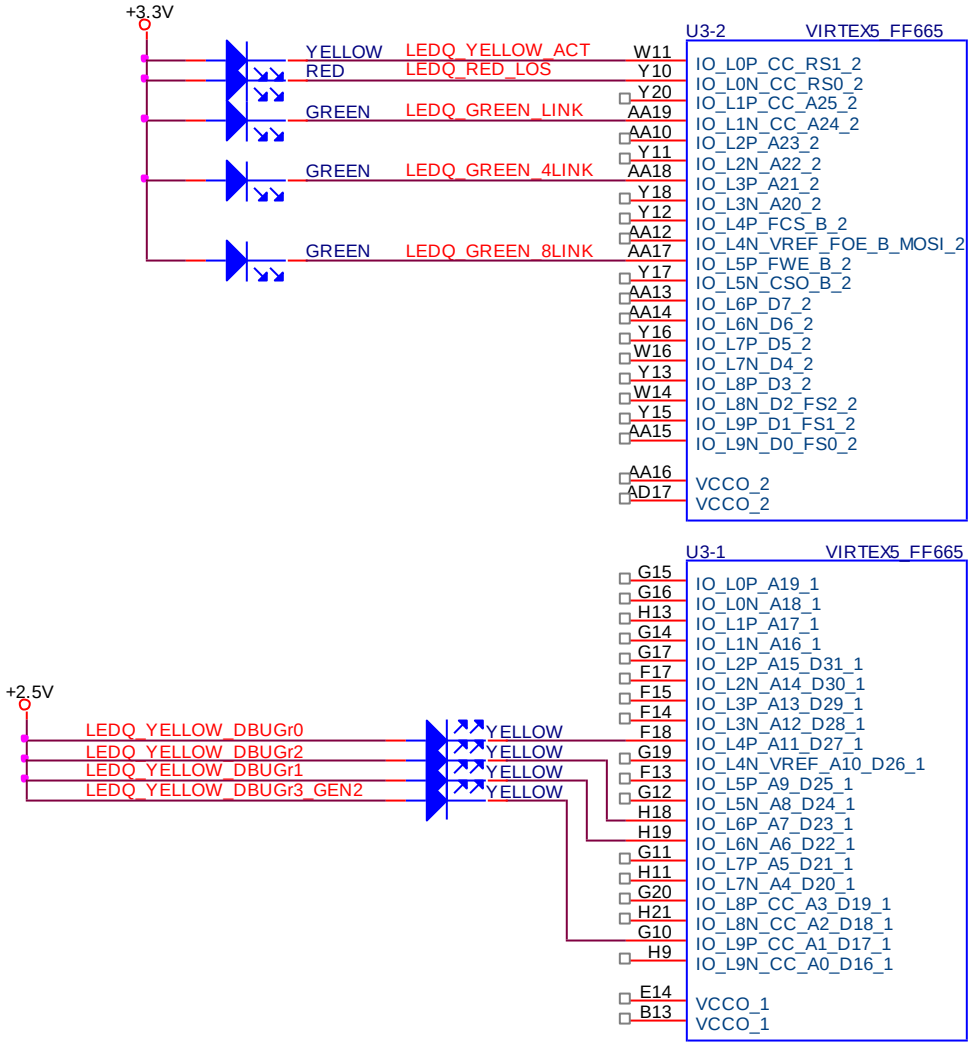


Figure 64 - FPGA Q LEDs

8.6 RS232

8.7 Synthesizer

9 PCI Express Interface

The DN9200K10PCIE8T can be installed in a PCIe slot. 16x or 8x slots are acceptable. The board will work in a 1x, 2x, or 4x slot, if you can physically manage to install them there (using an adapter such as the ones available from Catalyst enterprises).

The board can support 2.5Gb PCI Express 1.1-compliant signaling, or "Gen2" PCI Express 2.0 compliant signaling at 5.0 Gbs.

PCI Express interface is provided by FPGA Q, a Xilinx Virtex-5 LXT or FXT FPGA. For Gen 2 speeds, FX70T part is required.

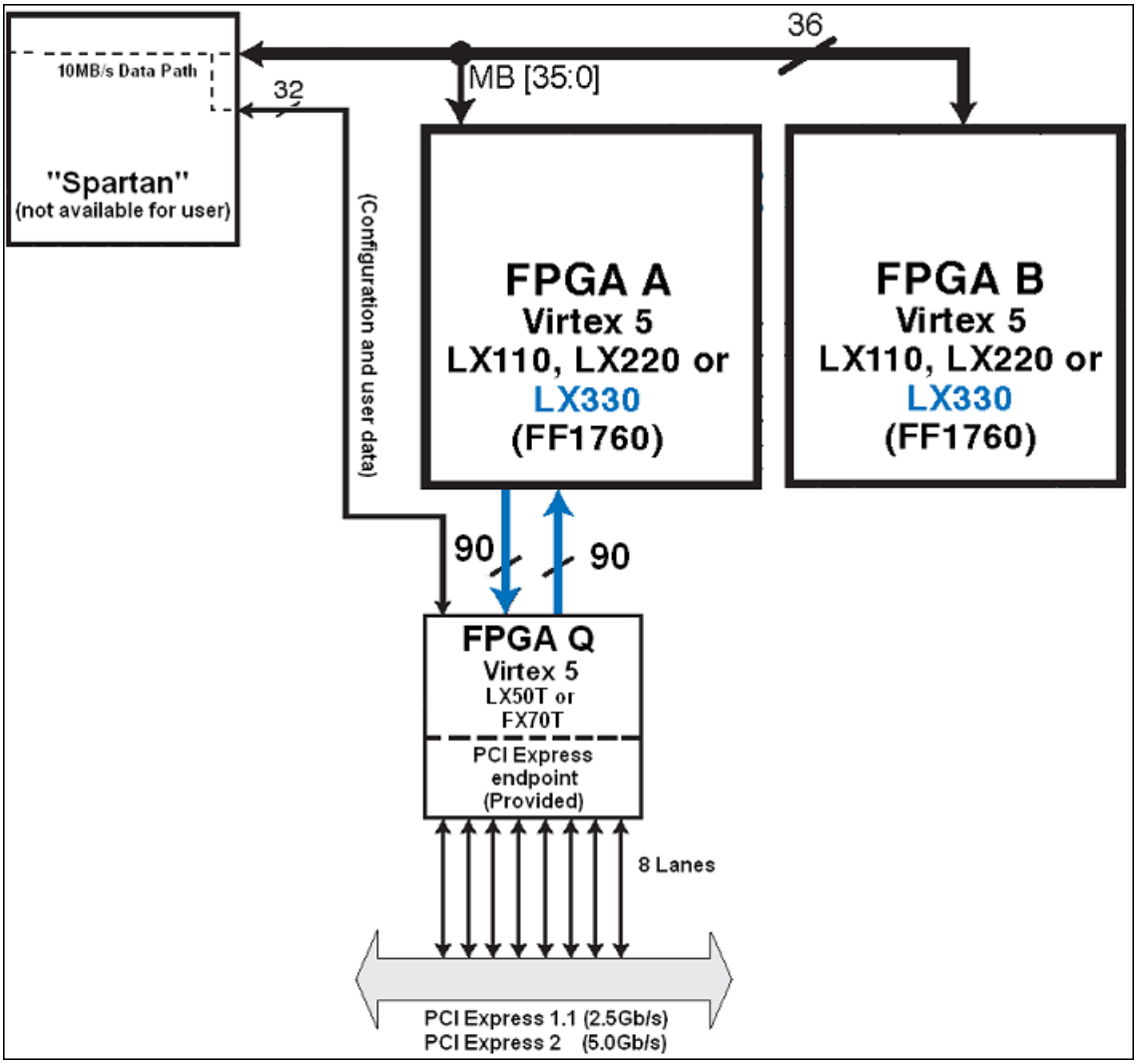


Figure 65 - PCI Express block diagram

Normally, a user will place his PCI Express endpoint IP in FPGA Q, and his high-density logic in FPGA A. A large amount of interconnect is provided between FPGA A and Q to easily keep up with a full-speed, 8-lane PCI Express endpoint.

The user can provide his own PCI Express IP, he can use the Xilinx PCI Express endpoint hard macro, or he can use the free, provided "full function PCI Express endpoint now with DMA™" core.

9.1 Host Interface, Electrical

The PCI Express signals from the host computer are connected directly to the LXT RocketIO IOs. As required by PCI express standard, the transmit signals (from the FPGA) are passed through ac-coupling capacitors. For fun, the receive signals (from the host) are also passed through ac-coupling capacitors.

The RocketIO requires a reference clock frequency to operate. On boards with an LX50T, this clock is provided on the MGTREFCLKP_112 pin at 100 MHz. As required by Xilinx, this frequency is identical to the frequency supplied by the host connector on the PCI Express REFCLK signal. On boards with an FX70T, the clock frequency is instead 250 MHz, exactly 2.5 times the frequency of the REFCLK signal provided by the host connector.

When creating a core using the Xilinx PCI Express core generator, you must tell the wizard program the frequency of this clock, and to which pins it connects.

There is also a Synthesizer that can generate 100 or 250 MHz for use with RocketIO. This synthesizer is described in another section. Xilinx does not recommend synthesizing a reference clock frequency for use with PCI Express because it is not a supported use model.

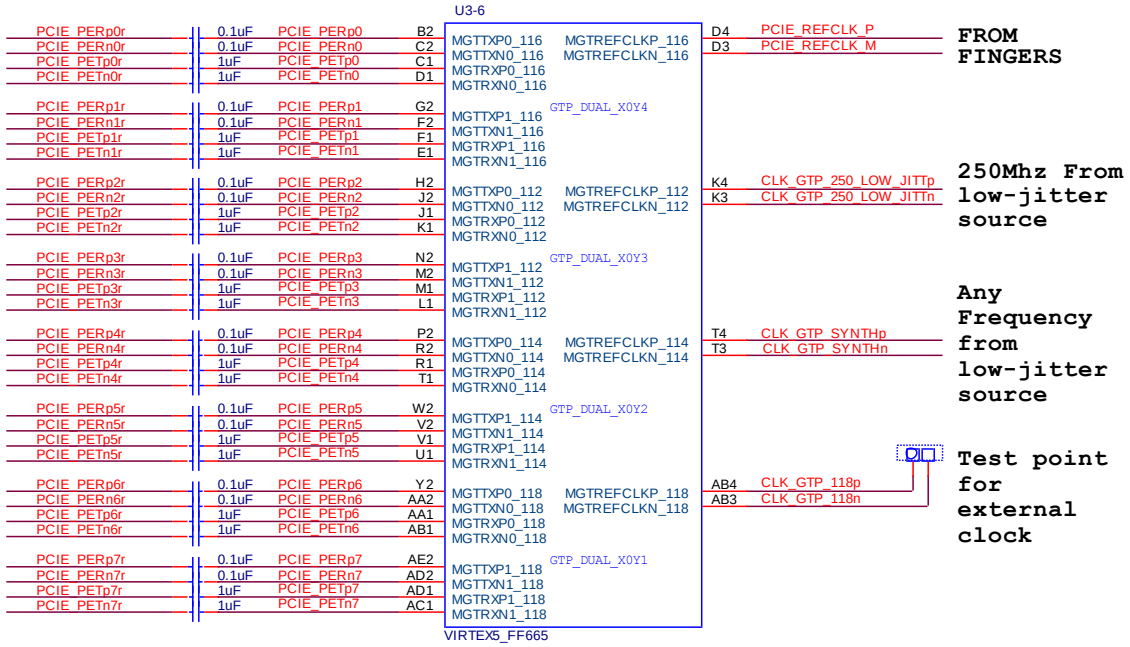


Figure 66 - PCI Express circuit

The order of the lanes is as shown above. Oh, also none of the lanes have inverted polarity, and you are required to support that if you are writing your own PCI Express endpoint.

We can run the PCI SIG electrical compliance test for you if you want.

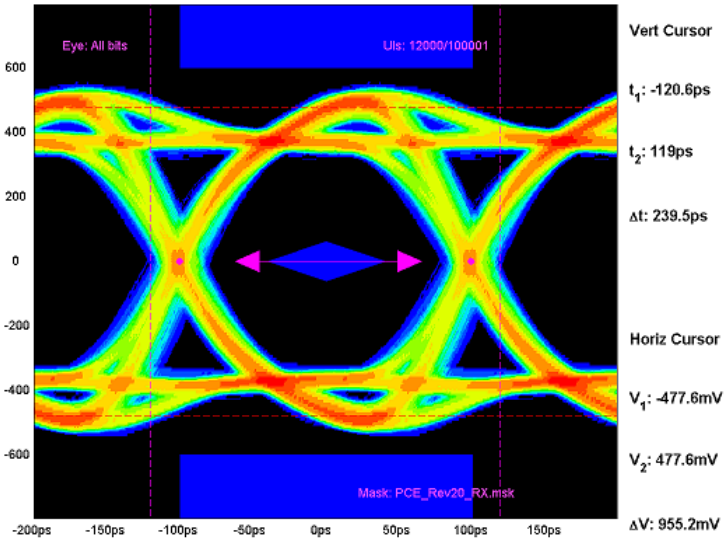


Figure 67 - PCI Express eye diagram

Here is the board installed with an FX70T passing the PCI Express electrical compliance test.

9.1.1 Power

The DN9200K10PCIE8T current capacity greatly exceeds the maximum allowed power requirements for a PCIe card (35W). As a result, the external power cable is required for operation, regardless of whether the board is installed into a PCI Express slot. The only voltage that is required for operation is 12V. All other voltages used on the board are regulated from this source.

The DN9200K10PCIE8T is designed to operate in hot-plug environments; however, most motherboards are not hot-plug capable. (They do not shut off 12V and 3.3V power signals when physical connections are lost). Therefore, a hot plug extender will be required for hot plug. Additionally we don't know how the provided "full function PCI Express endpoint now with DMA™" will behave, or how the Xilinx PCI Express endpoint hard macro will behave.

9.1.2 PCI-X

We assume you know the difference between PCIX and PCI Express. This board is designed to burst into flames when installed in a PCIX slot.

9.2 Host Interface, Mechanical

The form factor of the DN9200K10PCIE8T exceeds the allowable form factor for PCI Express in the vertical direction. This means that you will likely have to design the case for your system around the DN9200K10PCIE8T. Additionally, many "ATX" type computer cases do not fit the DN9200K10PCIE8T in the horizontal direction. If you are married to your computer case and motherboard, you can get one of these:

http://www.adexelec.com/pciexp.htm#PEX8LX

Otherwise, just get a case that fits the board.

9.3 Provided "Full-Function PCI Express endpoint"

Unless you need to prototype and test PCI Express logic, we recommend that you just use our provided PCI Express endpoint bit file. The provided bit file contains a high-speed implementation of the Xilinx PCI Express hard macro, adds a high-speed DMA engine, FPGA-initiated posting, implements high-speed IO between FPGAs A and Q at any frequency, allows PCI Express control of board functions such as configuration and clock settings, and comes with a working Windows and Linux driver. It will save you an approximate man month of work and writing and fully testing a custom implementation of a PCI Express Endpoint.

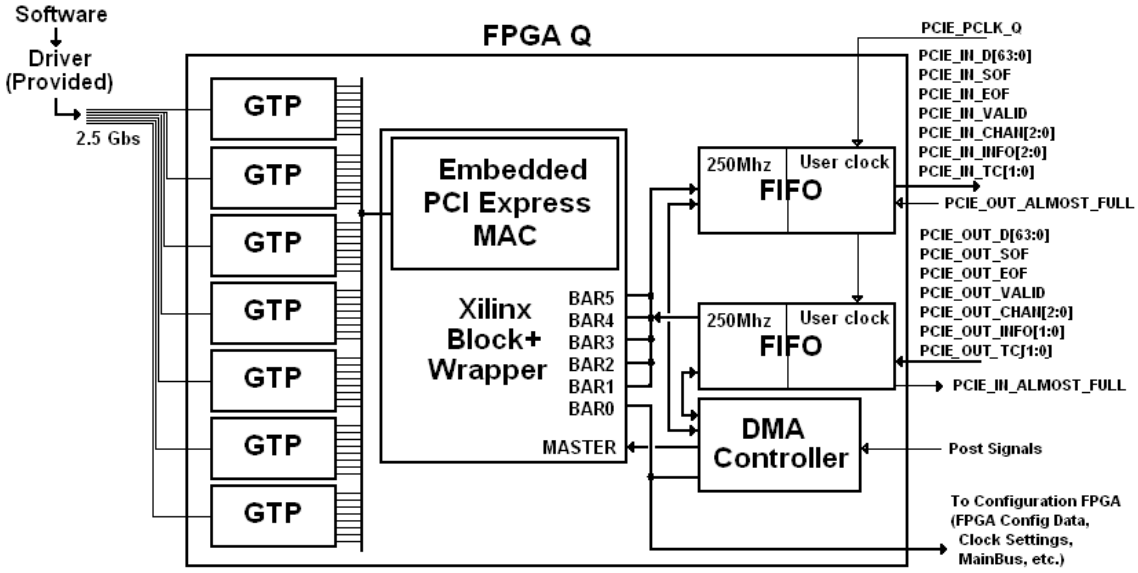


Figure 68 - Full function design block diagram

Access to FPGA A is through an allocation of memory space in the BAR regions of BAR2, 3, 4 and 5.

BAR0 is used for control of the DMA engine, for MainBus accesses to all FPGAs, and for board control and FPGA configuration.

Two DMA channels allow communication to FPGA using the full PCI Express bus bandwidth.

The best resource for using this endpoint (both from a host software and FPGA implementation standpoint) is the document provided at

FPGA_Reference_Designs\common\PCIE_x8_Interface\pcie8t_user_interface_manual.pdf

The BAR resources available are given below. These cannot be changed through any settings made available to the user.

Bar0: 0x0-0x1ff: PCI-E FPGA registers, rest is Configuration FPGA registers (8MB)
Bar1: 32-bit BAR, for User FPGA (8 MB)
Bar2-3: 64 bit BAR, for User FPGA (32MB)
Bar4-5: 64 bit BAR, for User FPGA (32MB)

By default, prefetch is turned off on 32-bit BARs
It may be on for the 64-bit bars.

The back end (FPGA A) interface is fixed at 64-bit.

In a 32-bit addressing machine, it will appear as if BAR2 is configured as a 32-bit bar and BAR3 will not be implemented. BAR4 will appear as a 32-bit bar, and BAR5 will not be implemented.

9.3.1 BAR 0 Access

The "Bar 0" accesses are reserved for board settings, FPGAs configuration and "Main Bus" communication. User-mode programs can access these registers to control the board from the PCI Host. Some of the useful offsets are given below:

Byte	Size	Name	Description
0x000	31:0	Version	Contains a version code for the firmware of LXT device (Read only)
0x008	31:0	ID	Always returns 0x4675_6C6C for "full function" design (Read only)
0x020	31:0	DMA0 Base Address	Lower 32 bit byte address of physical address where the DMA0 descriptor chain starts. This address must have the lower bytes cleared to match the DMA0 Address Mask register.
0x024	31:0	DMA0 Base Address	Upper 32 bits of Base Address [63:0], to form a 64 bit address. Set to 0 if using 32 bit addressing.
0x02C	31:0	DMA0 Control	
0x030	31:0	DMA0 Poll Immediate	
0x040	31:0	DMA1 Base Address	
0x04C	31:0	DMA1 Control	
0x050	31:0	DMA1 Poll Immediate	
0x98	357 Bytes	Scratch Pad	Read/Write space for user having fun and exercise imaginations
0x208	6:0	Config Control	Selects and FPGA and returns the value of these FPGA's PROG, INIT and DONE signals.
0x210	31:0	Config Data	Sends the given configuration word to the selected FPGA
0x238		FPGA Stuffing	
0x240		Main Bus ADDR	
0x248		MainBus Write	

0x250	MainBus Read
0x258	"Config Space" Write

9.3.2 BAR 1-5 Access

PCI Express reads and writes in the BAR1 – BAR5 memory space result in communication to FPGA A over the PCIE_IN* and PCIE_OUT* signals on FPGA A. This should be used in conjunction with the provided PCIe_interface module in FPGA A.

See source code here:
D:\FPGA_Reference_Designs\common\PCIE_x8_Interface

9.3.3 DMA Channels 0 and 1

There are two independent DMA controllers that are capable of descriptor chaining in the full-function endpoint. The register interface is described on the user CD in the documents at

D:\FPGA_Reference_Designs\common\PCIE_x8_Interface

It is best that you read the details there. Most users will not need to understand the control of DMA because the driver source code and binary in Windows and Linux is provided and works. There are two software interfaces to DMA.

9.3.3.1 Scatter/Gather

The DMA controller is capable of fetching descriptors from the host memory, allowing the DMA engine to follow scatter/gather chains.

The driver hooks for this weren't written yet when I wrote this. You might have to call for an update.

9.3.3.2 Large Buffers

In large buffers mode, the segment list is fixed and points to a ring of buffers in pre-allocated, locked driver memory space. The user has unsynchronized access functions that allow copying to and from these fixed buffers. The DMA engines loop around the fixed buffers constantly completing the DMA on the buffers. The user has access to controls that turn on and off the DMA when not in use.

Example use of this code is provided in the AETEST program in the file pcie_functions.cpp

9.3.4 DMA Posted Mode

Posted mode allows the FPGA A to initiate DMA transactions to and from the host memory space. This mode is possible using the Dini Group full-function DMA endpoint, but is not enabled in the user interface module due to lack of interest. Contact us to get access to posted mode.

9.3.5 DMA Main Bus

Main Bus is already pretty fast (100MB/s), however, if you really more over main bus then we can tell you how to do DMA on Main Bus. You might have to deal with synchronization issues on your own (read/write ordering).

9.3.6 Electrical

The electrical input and output characteristics are based on the PCI Express revision 1.1 and 2.0 requirements. The transmitted signal is slightly higher amplitude than that allowed by the specification in order to allow more flexible connection options (cabling or adapters) without compromising reliability. In addition, Pre-emphasis in the transceivers is set to ultra, which is not optimal, but will improve reliability in crappy systems.

If you need to pass PCI Express compliance electrical test with your board, please request the PCI Express compliance bit files from support. They are identical in function, but will pass compliance tests.

9.3.7 Timing

The provided module for FPGA A takes care of the external interface timing, so you can probably skip this section.

When using the full-function PCI Express endpoint, a source-synchronous communication technique is used between FPGA A and FPGA Q. Since the FPGAs both have zero-hold-time inputs, the optimal phase alignment between clock and data is when they are perfectly in phase. Therefore, the clock for FPGA A (PCIE_PCLK_A) is driven from the IOs of FPGA Q in the exact same manner as the IOs, and the clock for FPGA Q (PCIE_PCLK_Q) is driven from FPGA A in the exact same manner as the IOs. On the board, the data and clock lines are all phase-matched.

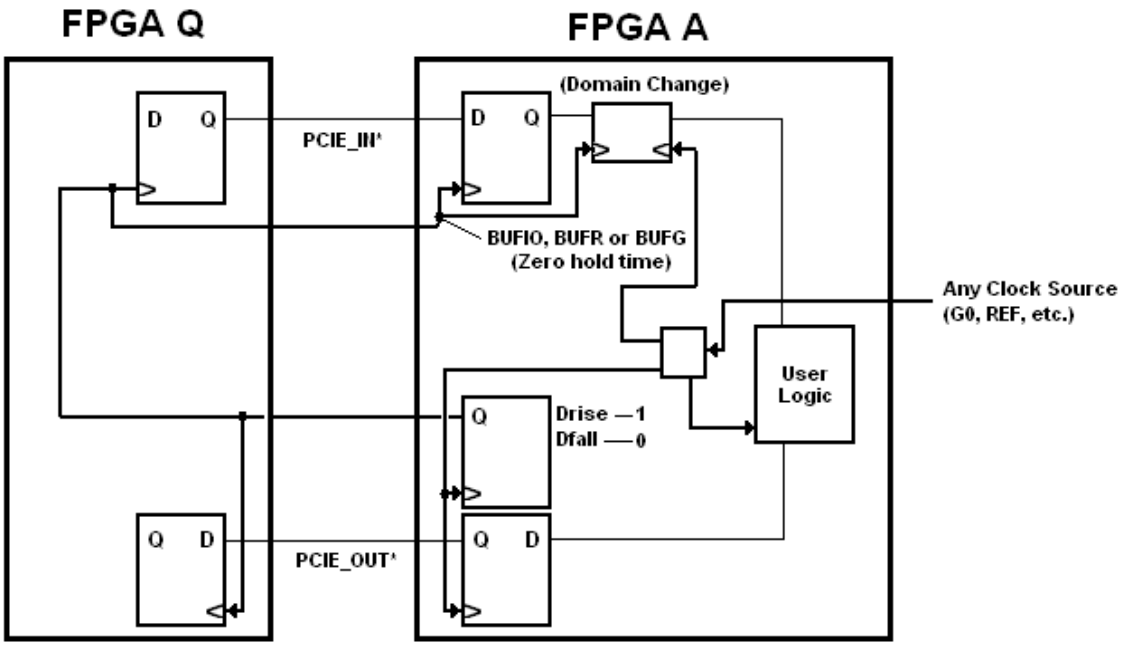


Figure 69 - FPGA A to Q clocking diagram

Driving clocks from IOs is best accomplished using a ODDR flip-flop. If you don't know what I am talking about, there is a description of exactly what to do elsewhere in this manual.

9.3.8 FPGA Interface

A Verilog module is provided that correctly implements the interface between the FPGA and the FPGA Q for PCI Express communication. The source for this module is provided on the user CD in the following location.

D:\FPGA_Reference_Designs\common\PCIE_x8_Interface\

A module, contained in the provided source file pcie_x8_user_interface.v, is an implementation of the interface that must be included in the FPGA A user design.

The user ("FPGA") interface presents 6 separate interface ports: Target Write, Target Read, DMA R0, DMA R1, DMA T0, and DMA T1. The Target Write and Target Read interfaces share BAR and address lines, as target reads and writes cannot happen simultaneously. Each interface has its own "enable", "accept", and "data" ports. Read interfaces also have a "data_valid" port. The "enable" signals are held active until the associated "accept" signal goes active. The "accept" signal for an interface may be tied high if it is guaranteed that transfers for that interface can be accepted every clock cycle (i.e. if the interface is connected to a block RAM). "Data_valid" can be pulsed with the "accept" signal, or any time after- this allows reads to be pipelined.

For the purposes of simulation, a model of low synthesizability of the LXT is provided.

9.3.8.1 LEDs

Six LEDs are controlled by the PCI Express FPGA:
Activity, Link1, Link4, Link8, and PERSTn, GEN2, and LOS

PERSTn directly shows the state of the PCI Express reset signal from the host. This is typically only during power-on. Activity is generated by the PCI Express FPGA whenever a packet is received. This signal on certain Intel-based hosts may blink constantly because of some mysterious configuration register read that gets generated all the time.

The Link1 LED will only be active when the PCI Express LED is communicating without error to a link partner, with a 1x negotiated lane width.

The Link4 LED will only be active when the PCI Express LED is communicating without error to a link partner, with a 4x negotiated lane width.

The Link8 LED will only be active when the PCI Express LED is communicating without error to a link partner, with a 8x negotiated lane width.

When the PCI Express LED has negotiated a 2x link, both Link 1 and Link8 will light. How did you manage to link in 2x mode? Send your interesting anecdotes to support@dinigroup.com

The LOS LED will light when there is no receiver detected on lane 0, or when some other thing isn't working.

Gen 2 will light if the design has linked at 5.0 Gbs.

9.3.8.2 FPGA-initiated DMA

The DMA controller is capable of issuing PCI Express transactions initiated from the FPGA A. This function is tested, but the interface is not documented. Contact us.

9.3.9 Host Interface, Software

Example software capable of configuring FPGAs, communicating over MainBus and DMA transfers to FPGA A is provided (AETest). You may wish to copy this code and use it as a starting point.

To communicate with the DN9200K10PCIE8T, you will need to find the device on the PCIe Bus with VendorID=17DF and DeviceID=1900. The device will register itself with the operating system as "Dini Group ASIC Emulator with Virtex 5 PCI Express" (OS dependant).

Note that many Dini Group products use this vendor and device ID, so differentiating between boards requires you to read at a minimum, the board type register and the board serial number register.

9.3.9.1 Driver

The source code for the DN9200K10PCIE8T's PCIe driver is provided.

Windows XP/Vista

Binaries for 32-bit windows, 64-bit windows (Itanium) and 64-bit windows (AMD/Pentium) are provided as a binary. Use the windows hardware manager to install these drivers. Source is provided, but shouldn't be required by most of you.

Linux

Source is provided for the linux driver. Compilation is probably required. (Provided binaries are unlikely to work). Also, source is only tested with the latest version of Linux, and may not be compatible with older version. To compile you will need the kernel source module installed on your computer. The executable created by the source is a kernel module which is loaded dynamically. A kernel module load script is provided.

DOS

Under DOS, only direct device access is supported. The DOS version of AETest program does not use a driver. You therefore need to figure out how to configure and access a device on the PCI subsystem. DMA is not supported.

Solaris

The Solaris driver does not support DMA.

9.3.9.2 Configuration Register writes

Board settings (clocks, FPGA temperatures, etc.) can be changed over PCIe by accessing the "Configuration Register" interface. A description of the registers in this interface is in the Configuration Section of this chapter.

Writes

To write to a configuration register, write to BAR0, offset 0x258. Send a 32-bit word of data. This data is encoded as follows

Bits 31-16: "Configuration Register" address in (only addresses 0xDF00-0xDFFF are valid. See the "Configuration Register" map in the "Configuration Section" section)

Bits 15-8: Ignored

Bits 7-0: The Data value to write to the register

Reads

To read from a configuration register, read one byte from PCIe at an address within Bar0, encoded as follows:

Bits 31-24: The DN9200K10PCIE8T's BAR0

Bits 23-16: the lower 8 bits of the address of the configuration register you would like to read (The upper 8 bits must be 0xDF, or the read will not be valid)

Bits 15-0: 0x0260

9.3.9.3 Main Bus

The Main Bus interface is how you can communicate to all FPGAs on the DN9200K10PCIE8T over PCIe (not just FPGA A). The bandwidth available over the Main Bus is much lower than that of PCIe, so performance is not as great using this method. For details about the Main Bus, see the Main Bus section in this chapter. Expected speeds will be 30 to 80 MB/sec.

To write to Main Bus over PCIe, write to BAR0 at the address QLPCI_REG_MBADDR with the 32-bit value representing the main bus address you would like to write to. Then, write a second PCIe write to address QLPCI_REG_MBWRDATA with 32-bit data representing the data that you would like to write to main bus. After the Spartan 3 has received a write to both the MBADDR and MBWRDATA registers, it will write to the main bus interface.

To read from the Main Bus over PCIe, first write to BAR0 address QLPCI_REG_MBADDR with the 32-bit value representing the main bus address you would like to read from. Then, read from BAR0, QLPCI_REG_MBRDDATA. The returned value will be the value read off the main bus at the selected address. When an error has occurred (No FPGA responded to the read request) the read will return the value 0xBBBBBBBB. If all you get is 0x1234567 this means the main bus is being used by USB at the moment.

QLPCI_REG_MBADDR	0x240
QLPCI_REG_MBCTRL	0x270
QLPCI_REG_MBWRDATA	0x248
QLPCI_REG_MBRDDATA	0x250

9.3.9.4 FPGA Configuration

The sequence required to configure FPGAs over PCI Express is given in the Configuration Section.

9.3.9.5 Direct PCIe to FPGA, DMA

Detail about the software required by the host of the DN9200K10PCIE8T can be found in

D:\ FPGA_Reference_Designs\common\PCIE_x8_Interface\pcie8t_user_interface_manual.pdf

This document should be used to design software to access the user design in FPGA A. DMA in particular requires accessing the either LX50T registers (in BAR0) to setup each transaction.

Using the device driver provided use the dma_scatter_gather_read() and dma_scatter_gather_write() functions.

Performance has been characterized using the DN9200K10PCIE8T reference design on Windows XP on a MSI MS6728 motherboard using the AETest application. The speeds are:

Read (DN9200K10PCIE8T to software): I have not yet performed this test.
Write (software to DN9200K10PCIE8T): I have not yet performed this test.

9.3.9.6 Direct PCIe to FPGA A, Target access

If DMA is not required, accessing FPGA A from the host software is super simple. Simply read or write to an address in BAR 1,2,3,4 or 5. In Linux this can be performed by mapping a page of memory in a user mode program to the physical address of a DN9200K10PCIE8T bar. In Windows driver, an IOCTL code is provided that will read and write individual bytes to the DN9200K10PCIE8T bar address range, or a block or memory.

9.3.9.7 Performance

Using the provided "Full function PCI Express endpoint now with DMA"™ the following speed measurements were taken:

DMA from host to FPGA[1]	510 MB/s
DMA from FPGA to host[1]	350 MB/s
Target access from host to FPGA[2]	66 MB/s
Target access from FPGA to host[2]	4 MB/s
Main Bus to FPGA from host[3]	11 MB/s
Main Bus from FPGA to host[3]	2.4 MB/s

Note 1: Using the "large buffers" DMA method in the driver. This method eliminates driver overhead.
Note 2: This speed can be increased by 2x using double-double word writes.
Note 3: This speed can be increased to the Target access speed in FIFO mode.

9.3.9.8 64-bit addressing

64-bit addressing has no effect on operation.

9.4 Other Provided Designs for the LXT

If you are not testing PCI Express endpoint logic specifically, you most likely want to use the provided "full function PCI Express endpoint now with DMA"™ design. Otherwise, you have the following options

9.4.1 No design

You can implement your design directly within the LXT, connecting directly to the Xilinx MGT. In this case, you will have to learn the peculiarities of the MGTs, and you will have to convert the output of the MGT into PIPE (fairly easy). You can also use the LXT as an additional FPGA in the case that you are not operating in a PCI Express slot at all.

9.4.2 PIPE

The "PIPE" bitfile provides the ability to have a standard, 125 MHz, 16-bit PIPE interface. Like the full-function design, you are required to use in FPGA a provided interface module. This module takes care of translating from the native GTP back end into a standard PIPE interface. It also takes care of external bus timing and clocking.

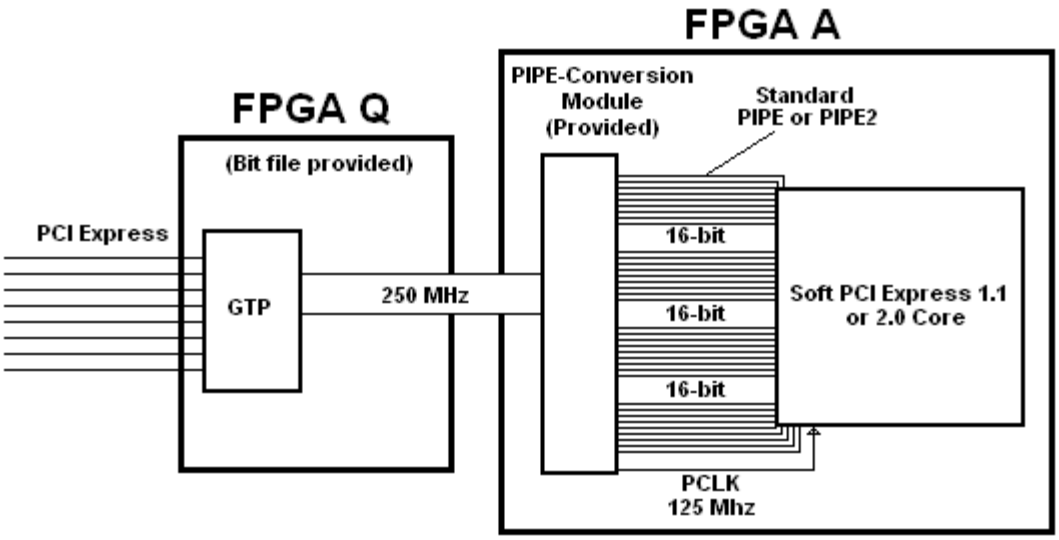


Figure 70 - PIPE design block diagram

We can also provide 8-bit, 250 MHz PIPE or PIPE that takes in an external clock. These modifications are not on the user CD but can be generated to suit your needs on request.

9.4.3 Slowdown PIPE Core

It can be challenging to place-and-route a PCI Express MAC in an FPGA which is capable of 8x operation and runs with a 125 MHz or even 250 MHz system clock. The PIPE slowdown core reduces the system clock "PCLK" frequency from full frequency to either 2, 4 or 8 times slower.

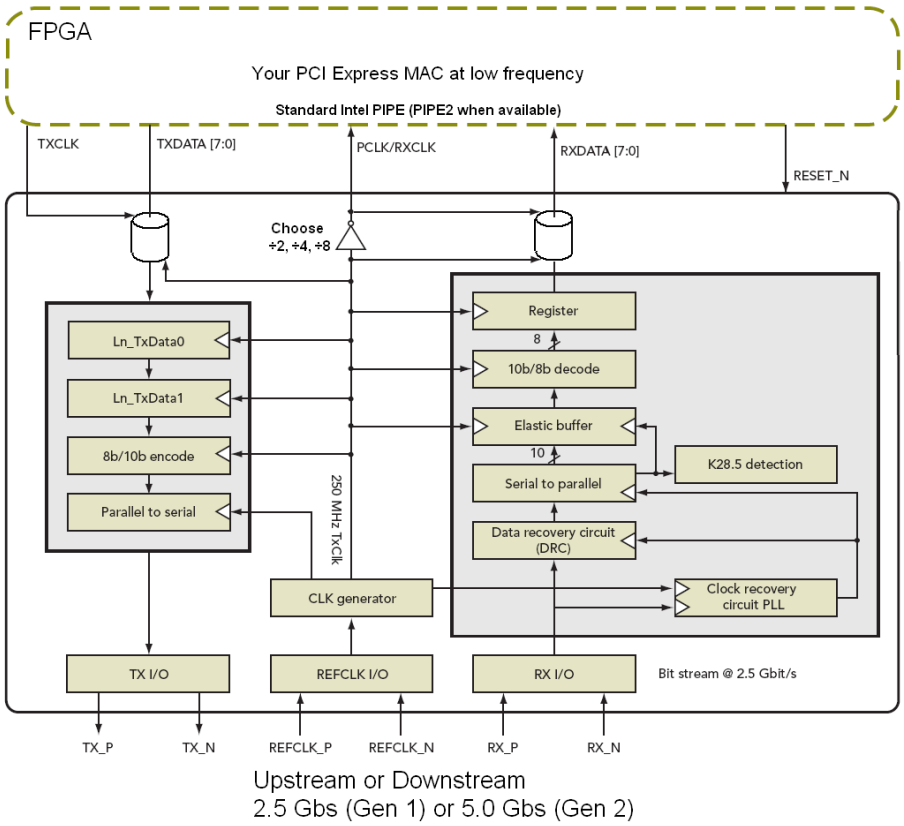


Figure 71 - PIPE Slowdown block diagram

Using this core, a PCI Express controller can interact with a real, full-speed link partner and test control paths that a non-interactive simulation might never test.

There is a fee for use of the PIPE slowdown core.

9.5 Troubleshooting

In PCI or PCI Express, when a bus master does not receive a responds for a read request within a certain timeout period, it will return 0xFFFFFFFF to the upstream requestor. This can happen for various reasons:

- The board has lost its configuration data (the PCI configuration space registers are not programmed)

- The FPGA on BAR

10Unusable pins

Some pins on the FPGA do not appear in the UCF file, and are not usable by the FPGAs.

10.1 Adjacent RocketIO

FPGA Q has some IO pins that are connected directly to ground. These pins are AA5, AB5, AF4, AF3, A3, B4, B5, D5, and E5. It is recommended that you drive these pins with a constant low value, and assign a high drive-strength driver to the IO type. These pins are intended to help shield the sensitive RocketIO power supply pins from IO switching noise.

10.2 No Connect

10.3 Configuration

The following pins (All FPGAs) are the SelectMap data pins, used to configure the FPGAs. These pins are connected to both Virtex-5 FPGAs. Using these signals for FPGA interconnect is possible, but may interfere with the configuration circuitry on the DN9200K10PCIE8T.

10.4 VREF/DCI

If you try to use a pin reserved for DCI calibration or a VREF reference voltage, then the tool will not let you complete the place-and-route.

11System Monitor/ADC

The new Virtex 5 feature System Monitor allows the FPGA to use some of its IO as analog-to-digital inputs.

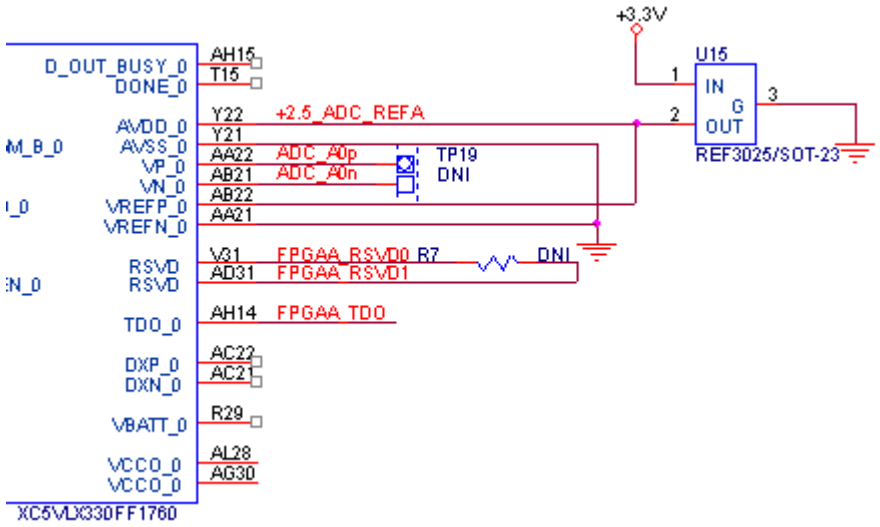


Figure 72 - Sysytem monitor circuit

The voltage measurements at these inputs are referenced to the voltage on the pin VREFP. On the DN9200K10PCIE8T, this voltage is generated by a high-precision external voltage reference IC.

The primary ADC input is routed to a differential test point. There is one test point labeled "ADC" for each FPGA.

12Reset

There are two reset circuits on the DN9200K10PCIE8T. One is the power-on reset, or "Hard Reset", that holds the board, including the configuration circuitry in reset until all power supplies on the board are within their tolerances. The second reset circuit is the user reset, "FPGA reset", "user button" or "Soft reset".

12.1 Power Reset

The power-reset signal holds the configuration circuit (including a micro controller and Spartan 3 FPGA) in reset. It also causes the FPGAs to become un-configured, and causes the RSTn signal on the daughtercards to be asserted. When the board is "in reset", the "Hard Reset" LED, DS20, is lit red. It is located about an inch above the USB connector.

When the board is in reset, FPGAs cannot be configured, USB does not function (the host computer will not be able to communicate with the device), PCIe cannot access the FPGA or configuration functions (the device will still be accessible from PCIe, and LX50T registers can still be read and written). When in reset, the Spartan configuration FPGA remains configured, but all of the logic in the device is cleared.

Pressing the "HARD RESET" button, S1, located near the ATX power connector, can trigger the Power reset. This reset cannot be triggered over PCI Express or USB. It is also triggered with one or more voltages on the board fall below, or above a certain threshold. These thresholds are given below:

Voltage	**Min**	**Max**
1.0V (A):	0.94V	1.1V
1.0V (B):	0.94V	1.1V
1.8V:	1.67V	3.8V
3.3V:	2.7V	3.8V
5.0V:	4.0V	5.6V
12V:	--	--
2.5V	2.25V	2.7V

When the board comes out of reset, the micro controller goes through an initialization process that will cause all current settings to be lost, including clock settings. Also, the configuration circuit will act as if the board has just powered on and read from the main.txt file to configure FPGAs.

When reset is triggered, it remains triggered until 55us after all trigger conditions are removed. This behavior prevents USB from behaving in such a way to permanently disable USB on the host machine.

12.2 User Reset

The "USER RESET" circuit is intended for use by the user. When this reset is asserted, the RESET_*# signal (from the schematic), is asserted to each FPGA. After at least 200ns, this signal is de-asserted simultaneously to each FPGA. This signal is connected to a regular user IO on the FPGA, so it is up to the FPGA designer to implement reset correctly within his design.

The User Reset is asserted whenever the "User Reset" button is pressed. This button, S2, is located just above the USB connector. There is no LED indicating the state of user reset. User reset is also asserted when the reset vendor request is sent over USB.

When User reset is asserted, the RSTn signal to each daughtercard is also asserted.

The arrival time of the assertion and de-assertion of reset is the same at all FPGA inputs. Additionally, the reset signal is timed such that it can be sampled synchronous to CLK_MB48.

13JTAG

There are two JTAG headers on the DN9200K10PCIE8T. The first, J6, is used only to update the board's firmware. The second, J5 is connected to the JTAG port of the Virtex-5 FPGAs. This interface can be used for configuring the FPGAs, or using debugging tools like ChipScope or Identify.

13.1 FPGA JTAG

The connector for FPGA JTAG is shown below.

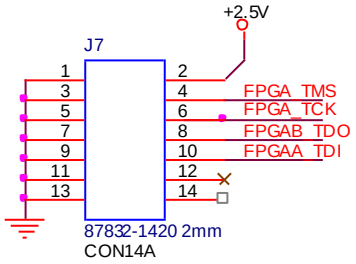


Figure 73 - FPGA JTAG circuit

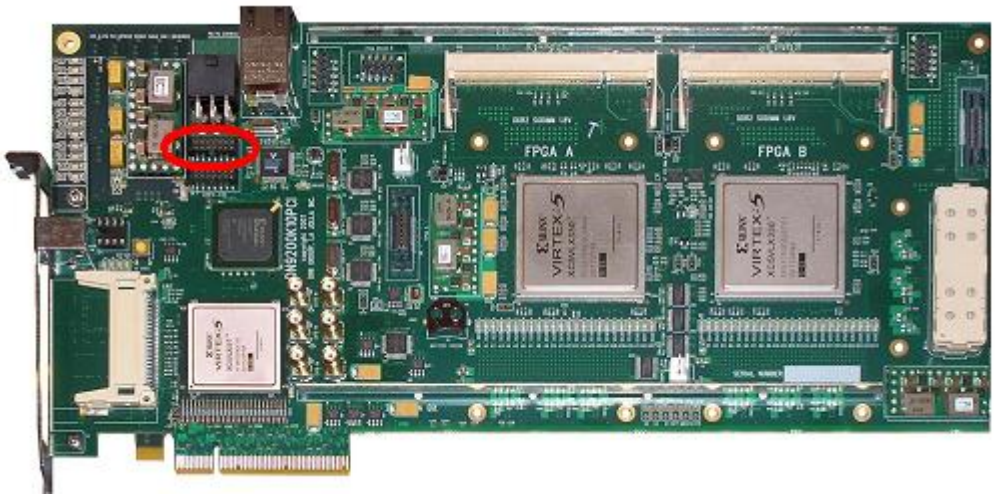


Figure 74 - FPGA JTAG locator

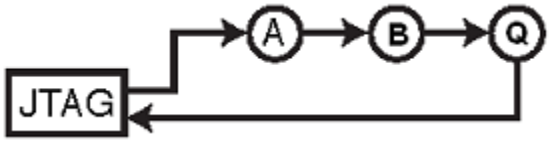


Figure 75 - FPGA JTAG block diagram

Note that the signal "TDO" on the header and in the schematic refers to the "TDO" port of the FPGA, not the connector.

The order of the FPGA JTAG chain is FPGA A->FPGA B->FPGA Q. There are no other components in the chain. If you received your board with fewer than two FPGAs installed, then the chain will be shorter.

The voltage of the JTAG chain is fixed at 2.5V and cannot change. Hot-plug on this header is allowed. The header is a 2mm pin grid dual row with shroud and polarization key.

13.1.1 Compatible Configuration Devices

The JTAG header is designed to work with the Xilinx Platform USB cable. The JTAG chain is tested at manufacture using a Platform USB cable at 12 MHz.

The driver installation process for the Platform USB cable is relatively difficult for a USB device. Follow the instructions carefully.

In order to achieve high-speed configuration using a Parallel IV cable, you need to enable ECP mode on your parallel port. This is probably a BIOS setting on your computer.

13.1.2 ChipScope

In order to use JTAG debugging tools on the DN9200K10PCIE8T, you do not need to configure via JTAG.

13.2 Firmware Update Header

The firmware update JTAG header J6, should not be used unless you are updating the DN9200K10PCIE8T firmware. This header is used with a Xilinx Platform USB or Parallel IV cable. The instructions for updating the firmware are in the Controller software chapter.

13.3 Troubleshooting

If you are having problems getting JTAG to work, try connecting the Xilinx Platform USB cable to the JTAG header and running the Xilinx program iMPACT. iMPACT will generate a failure log that you can email to support@dinigroup.com. If you have an upgraded board, please mention this in your email.

14RS232 Interface

RS232 access is available to all FPGAs through the header P4 "FPGA RS232". To connect to this header, use the provided .1" header-to-DB9 cable to connect to a PC's serial port.

The TX and RX signals use the RS232 data protocol, so the FPGA will have to implement a UART in its logic.

All FPGA share the same RX and TX signals, so only one FPGA should use the interface at a time. RS232 requires a 12V to -12V signaling level, which is not available on Virtex5 FPGAs, so an external RS232 buffer is used.

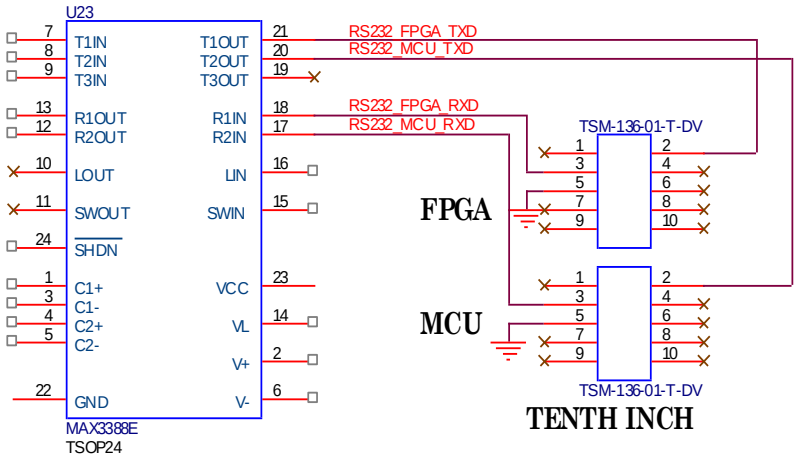


Figure 76 - RS232 circuit

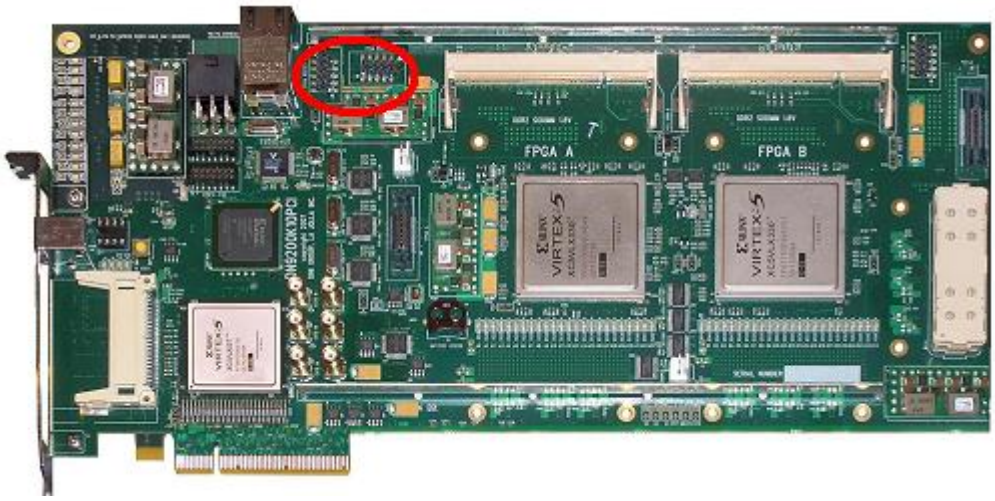


Figure 77 - RS232 locator

One the board, pin 1 is marked with a big, unmistakable, white circle dot. On the provided cable, pin one is marked with a red stripe on the cable. Hot-plugging this connector is acceptable and encouraged.

The port settings required on the serial ("COM") port of your computer are dependent on the UART in the FPGA. Since the flow-control signals on the serial cable are not connected to the FPGA, you cannot use "hardware handshaking".

The other port settings: software flow control, parity, stop bits, speed and data bits are user design dependent. There is no provided RS232 reference design.

14.1.1 Configuration RS232

A second RS232 header (P3) is for the configuration circuitry to give feedback to the user. It is described in the section "Configuration Section".

15Temperature Sensors

Each FPGA is connected to a temperature monitor. This monitor can internally measure the temperature of the FPGA silicon die. The maximum recommended operating temperature of the FPGA is 85°. The accuracy of the temperature sensor is about +0C° to +5C°. When the configuration circuitry measures the temperature of any FPGA rise above 80°, it will immediately un-configure the hot FPGA, and prevent it from re-configuring. When the temperature drops below 80, the configuration circuitry will again allow the FPGA to configure.

When this occurs a message will appear on the CONFIG RS232 port (P3). An example test output is given below.

****

TEMPERATURE ALERT: FPGA A

CURRENT TEMPERATURE: 81 DEGREES C
THRESHOLD TEMPERATURE: 80 DEGREES C
THE FPGA IS BEING CLEARED IN AN ATTEMPT TO PREVENT HEAT DAMAGE.
SOFTWARE WILL PREVENT RECONFIGURATION UNTIL THE TEMPERATURE
DROPS A FULL DEGREE BELOW THE THRESHOLD TEMPERATURE.
****
****
TEMPERATURE ALERT: FPGA A
CURRENT TEMPERATURE: 79 DEGREES C
THRESHOLD TEMPERATURE: 80 DEGREES C
THE FPGA HAS DROPPED BELOW THE ALARM THRESHOLD
AND MAY NOW BE RECONFIGURED.
****

The FPGA operate as hot as 120°C before melting, ejecting hot acid on your hand, but at temperatures above 80°C, logical operation is not guaranteed. You can use the temperature setting in the ISE place and route tool to make timing allowances for operating the FPGA out-of-range. If you want to disable the temperature limit on the DN9200K10PCIE8T, you can do that using a menu option in the configuration RS232 interface. You can also increase the maximum temperature allowed.

On designs with pathologically noisy IO, there is a significant "ground bounce" effect in the FPGA, and the temperature sensors can have errors as high as 30 C°. To correct this you can

- Increase temperature threshold to 100°C. (Adjusting timing in ISE)
- Reduce IO frequency to below 150 MHz
- Follow the Xilinx SSO limits on IOs
- Use LVDS IO

16Encryption Battery

The Virtex5 FPGA supports bit stream encryption. When using encryption, the FPGA must decode the bitstream using a secret key that is stored in a persistent memory in the FPGA. When the DN9200K10PCIE8T is powered off, a voltage is supplied to the FPGA by a battery installed in socket X2.

X2 is designed to house a CR1220-type lithium coin-cell battery. Typically, these batteries produce 3.0V. The socket may also work with battery types DB-T13, L04, PA. These however, have not been tested. Insert the battery positive side up.

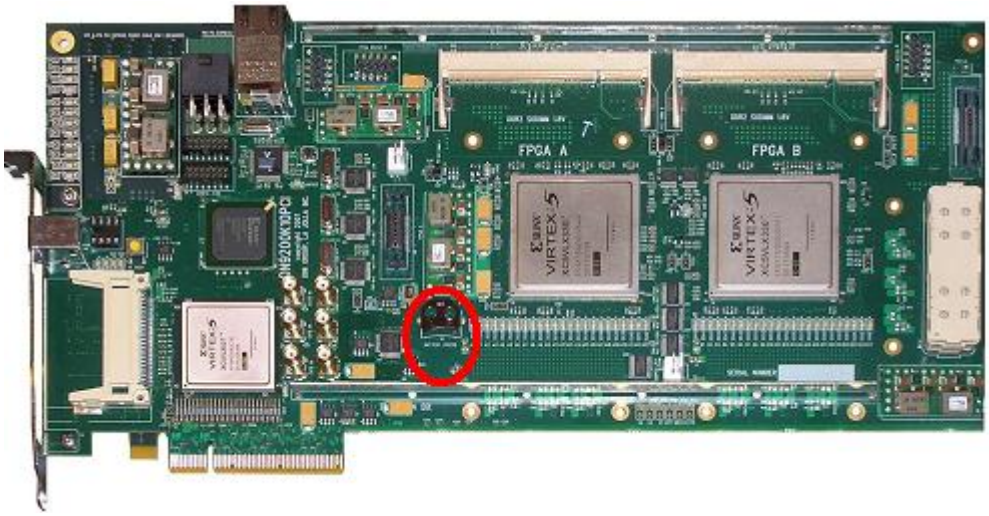


Figure 78 - Battery locator

The same battery is used for both FPGAs. Removing the battery will cause the FPGAs to lose their encryption memories, and will have to be re-programmed before they can work with encrypted bitfiles again.

To create encrypted bitfiles, turn on the "encryption" option in bitgen. The program will produce an additional output file with an .nky extension. Use the program iMPACT with a Platform USB JTAG cable (plugged into the FPGA JTAG connector on the DN9200K10PCIE8T) to load this .nky file into each FPGA.

When using a bitfile with encryption enabled, the DN9200K10PCIE8T will not be able to read the FPGA type out of the bitstream. It will therefore prevent your FPGA design from loading into the FPGA. To disable this behavior, you must disable sanity check. Adding the following line to your main.txt file can do this

```
Sanity check: n
```

Also, when using encryption, you must be careful to correctly set the "startup clock" option correctly in bitgen, or the FPGA will fail to configure, and won't tell you why.

Whatever you do, if you love your FPGAs, do not disable the "CRC Check" option in bitgen. This option was originally called "Do you want your FPGAs to *not* catch on fire?"

16.1 External Battery

Normally, swapping the battery without losing the encryption data requires having the board powered on while changing the battery. This is tricky.

In order to allow the swapping of a battery with the board powered off, there is a test point connected to the battery power that can be used to attach an external battery or voltage source.

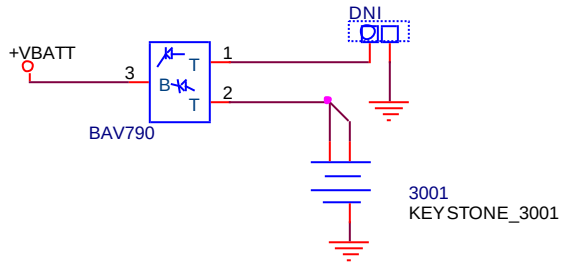


Figure 79 - battery circuit

17LED Interface

This section lists all of the LEDs. More detailed explanations of the LED functions may be in the sections describing the board system that contains the LED.

17.1 Configuration Section LEDs

These LEDs are controlled by the board (User has no control).

Reference	Name	Color	"ON" Condition
Power LEDs			
DS9	+1VA	RED	+1.0V on FPGA A has failed
DS12	+1VB	RED	+1.0V on FPGA B has failed
DS13	+DIMM_A	RED	Voltage on DIMM A has failed
DS14	+DIMM_B	RED	Voltage on DIMM B has failed
DS15	+2.5V	RED	+2.5V has failed
DS16	+3.3V	RED	+3.3V has failed
DS17	+5.0V	RED	+5.0V has failed
DS10	WARN DIMM A	RED	Voltage on DIMM A is not 1.8V
DS11	WARN DIMM B	RED	Voltage on DIMM B is not 1.8V
DS1	POWER WARN	RED	You didn't connect power cable
DS18	ON	GREEN	Always when board is on
DS20	SYS RESET	RED	Board is stuck in reset
Configuration Status LEDs			
DS23	ERRCONFIG	RED	An FPGA has failed to configure
DS24	ERRTEMP	RED	An FPGA has overheated
DS27	MB ACT	YELLOW	MainBus has activity
DS28	USB ACT	YELLOW	MainBus has activity over USB
DS90	PCI ACT	YELLOW	MainBus has activity over PCIE
DS89	CFACT	YELLOW	CompactFlash card is being read
DS30,DS31, DS32,DS33	MEANINGLESS	RED	You least expect it
DS35,DS36, DS37,DS38	MEANINGLESS	GREEN	You least expect it

DS88	FPGA_Q_LOL	RED	GTP clock synthesizer failed
DS22	G0_LOL	RED	CLK_G0 synthesizer failed
DS25	G2_LOL	RED	CLK_G1 synthesizer failed
DS29	G1_LOL	RED	CLK_G2 synthesizer failed
DS19	A DONE	BLUE	FPGA A is configured
DS26	B DONE	BLUE	FPGA B is configured
DS34	SPARTAN_DONE	BLUE	Spartan is configured (always on!)
DS87	Q DONE	BLUE	FPGA Q is configured

PCI Express status LEDs

DS7	PCIE GEN2	YELLOW	PCI Express is linked at 5Gbs
DS4	LINK1	GREEN	PCI Express is linked with 1 lane
DS6	LINK4	GREEN	PCI Express is linked with 4 lanes
DS5	LINK8	GREEN	PCI Express is linked with 8 lanes
DS3	PCIE LOS	RED	PCI Express could not link
DS8	PCIE_PERSTn	RED	PCI Express is reset by host
DS2	PCIE ACT	YELLOW	PCI Express is in use
DS91,DS92, DS93	PCIE DEBUG	YELLOW	General Purpose LED for FPGA Q

17.2 User LEDs

These LEDs are connected to an FPGA and are controller by the user. The meaning of the LED is design-dependent. Below is the general circuit used to connect user LEDs. To turn the LED on, drive the signal low. To turn off, tri-state or drive-high the signal.

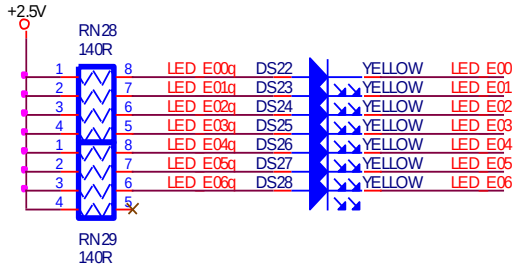


Figure 80 - LED circuit

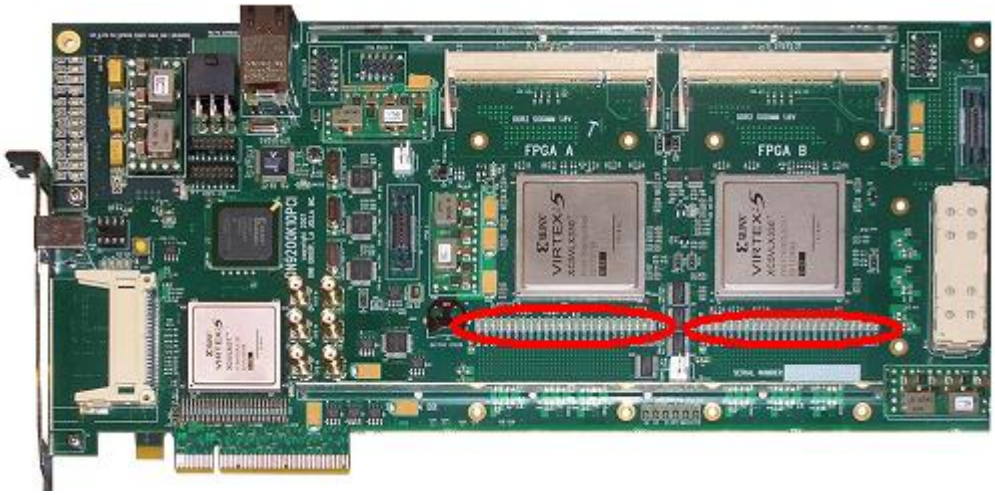


Figure 81 - LED locator

The user LEDs are connected to banks where the daughtercards are connected. The "Bank Voltage" may not match the LED's current source voltage. In this case, use the drive standard corresponding to the bank, and not the LED. For example, when a LVCMOS25 daughtercard is attached and all other signals on the bank are using the LVCMOS25 standard, use the LVCMOS25 standard for the LED on that bank. Do not use DCI on LED signals. You can control the brightness of LEDs by either using a low-drive setting (DRIVE=2ma in the .ucf file), or by making the output bounce rapidly high and low like my cat.

Part Reference	**LED Name**	**Color**
DS39,DS40,DS41,DS42, DS43,DS44,DS45,DS46, DS47,DS48,DS49,DS50, DS51,DS52,DS53,DS54	USER LEDs (FPGA A)	YELLOW
DS59,DS60,DS61,DS62	USER LEDs (FPGA A)	RED
DS55,DS56,DS57,DS58	USER LEDS (FPGA A)	GREEN
DS63,DS64,DS65,DS66, DS67,DS68,DS69,DS70, DS71,DS72,DS73,DS74, DS75,DS76,DS77,DS78	USER LEDs (FPGA B)	YELLOW
DS83,DS84,DS85,DS86	USER LEDs (FPGA B)	RED
DS79,DS80,DS81,DS82	USER LEDs (FPGA B)	GREEN
T1	Ethernet LINK1000	GREEN
T1	Ethernet Activity	YELLOW
DS21	Ethernet LINK100	GREEN

FPGA A and B each have a total of 24 user-access LEDs. The LEDs are numbered 0 to 23. The location of the IOs to use for these LEDs can be found in the provided UCF file or the netlist. The name of each LED is labeled in silkscreen next to the LED.

17.3 Ethernet LEDs

These LEDs are controlled by the Ethernet PHYs connected to FPGA B. They can also be user-controller by setting registers in the serial interface of the PHYs.

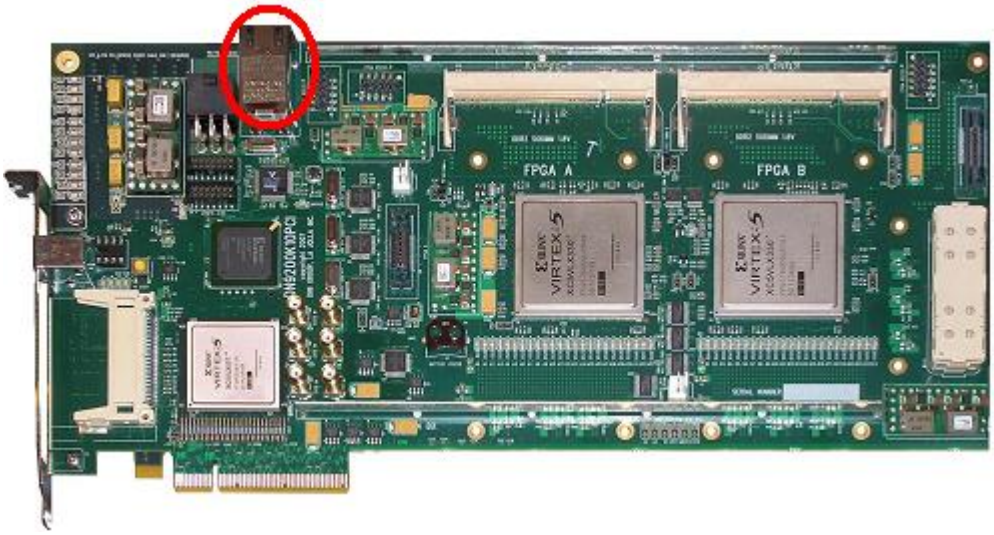


Figure 82 - Ethernet locator

T1 and T2 are the RJ45 jacks on the top edge of the board. There is a yellow and a green LED embedded in this connector, facing the board edge.

17.4 Power LEDs

These LEDs indicate is one or more power supplies fail, either outputting a voltage that is too high or too low. The voltage that the LED indicates is marked in silkscreen near the LED.

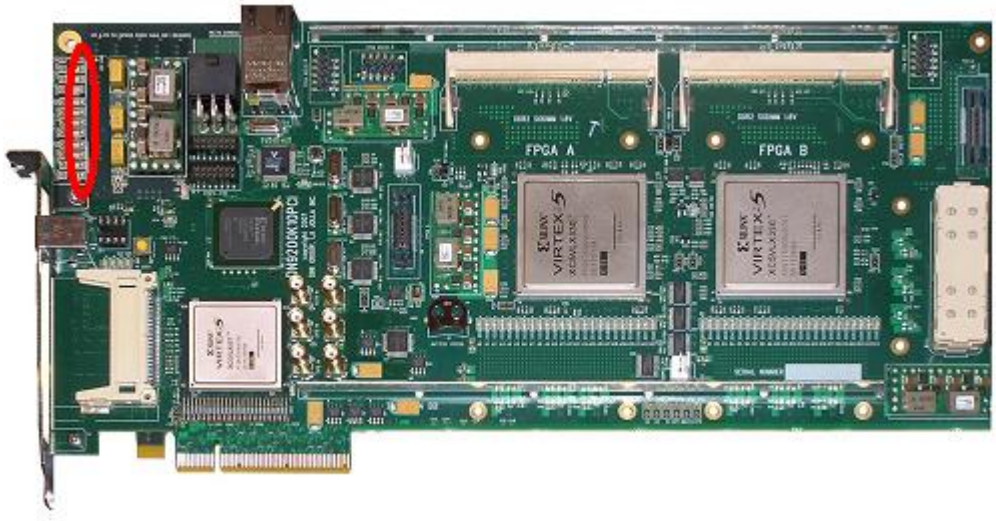


Figure 83 - Power fail LED locator

17.5 Unused LEDs

These LEDs are controlled by the configuration circuitry. One GREEN LED is always on. One yellow one flickers when something undefined is happening. Two RED ones signal which FPGA is undergoing some sort of configuration operation, and will pause with that indication if there is an error.

The primary purpose of these LEDs if for Dini Group to debug its software, so I wouldn't be surprised if this information was outdated already.

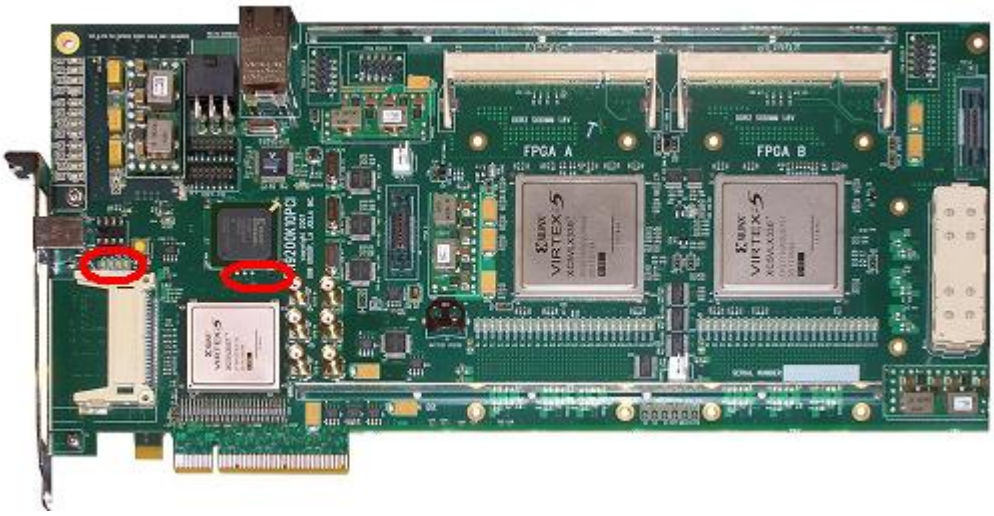


Figure 84 - Unused LED locator

18DDR2 DIMM Sockets

There are two "DDR2" memory socket interfaces on the DN9200K10PCIE8T.. By convention, the name of this interface connected to FPGA A is DIMMA, the one connected to FPGA B is DIMMB. In this section, the interfaces may be called "DIMM", "SODIMM" or "DDR2" interface interchangeably.

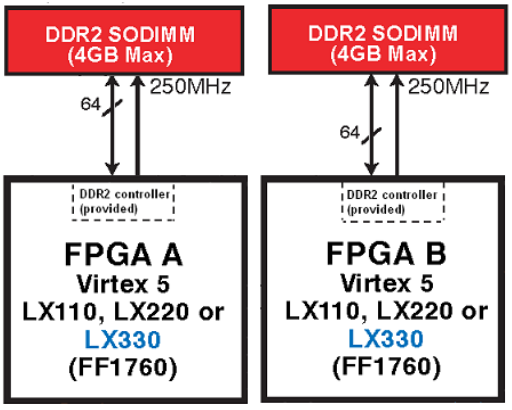


Figure 85 - DIMM block diagram

Signal names given in this section, and in other documentation (ucf files) are given in the form DIMMB_<signal name>.

18.1 Power

Each DIMM and its associated FPGA bank receives current from a dedicated adjustable power supply. Each DDR2 SODIMM is capable of drawing 5A of current when in continuous auto-precharge mode.

The DN9200K10PCIE8T is capable of providing this amount of current.

18.1.1 Interface Voltages

The "standard' DDR2 interface voltage is +1.8V. The banks that connect to the DIMM interface are powered by 1.8V, and the power pins on the socket is connected to this same power net. In a DDR2 interface, most of the DIMM signals are driven using the SSTL18_DCI drive standard

DIMM_A*	SSTL18_I
DIMM_CAS#	SSTL18_I
DIMM_RAS#	SSTL18_I
DIMM_BA*	SSTL18_I
DIMM_WE	SSTL18_I
DIMM_ODT*	SSTL18_I
DIMM_CSE*	SSTL18_I
DIMM_S*	SSTL18_I
DIMM_DQS*P	DIFF_SSTL18_II_DCI
DIMM_DQS*N	DIFF_SSTL18_II_DCI
CLK_DIMM_CK*P	DIFF_SSTL18_I
CLK_DIMM_CK*N	DIFF_SSTL18_I
CLK_DIMM_CK2P	LVDS_EXT
CLK_DIMM_CK2N	LVDS_EXT

DIMM_DQ*	SSTL18_II_DCI
DIMM_DM	SSTL18_II_DCI
DIMM_SDA	SSTL2_I_DCI or LVDS
DIMM_SCL	SSTL2_I_DCI or LVDS
DIMM_DQ64	SSTL18_I and SSTL18_I_DCI

The DIMM interfaces are not designed for hot-plug.

The CLK_DIMM_CK2P/N signal is intended to be driven from the FPGA (at 1.8V) into the FPGA (at 2.5V). Its arrival at the FPGA and the arrival of CLK_DIMM_CK0 and CLK_DIMM_CK1 at the SODIMM module are synchronized. It can be used as a feedback clock for a PLL, or as a primary clock for the DIMM interface.

The DIMM_DQ64 is length-matched to the other DQ* signals. It has no known purpose.

18.1.2 Changing the DIMM voltage

If you need to change the voltage of the DIMM interface, there is a set of jumper points provided for each interface allowing power to be regulated at a different voltage. The jumper has four settings:

PIN 1 – PIN 2	DIMM Voltage is 3.3V
PIN 3 – PIN 4	DIMM Voltage is 2.5V
PIN 5 – PIN 6	DIMM Voltage is 1.8V
NO JUMPER	DIMM Voltage is 1.5V

Any other combination of jumpers produces some other voltage that is too high for the FPGA to handle.

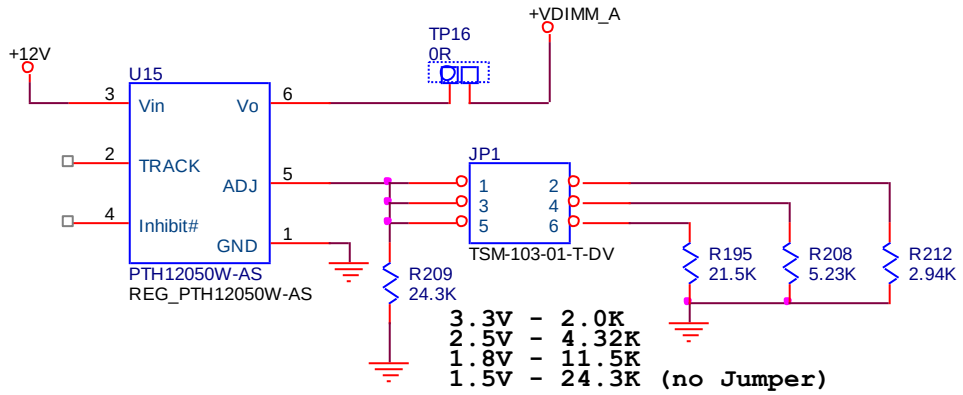


Figure 86 - DIMM Voltage selection circuit

If you are interested, you can see how the jumpers affect the voltage output of the regulator. If you want to store he jumper (when in 1.5V mode), you could safely do that by connecting the jumper PIN 1 – PIN 3 or PIN 2 – PIN 4 or something.

Some Dini Group SODIMMs requires these strange power supply voltage. (DNSODM_SDR, DNSODM_DDR1, DNSODM_DDR3).

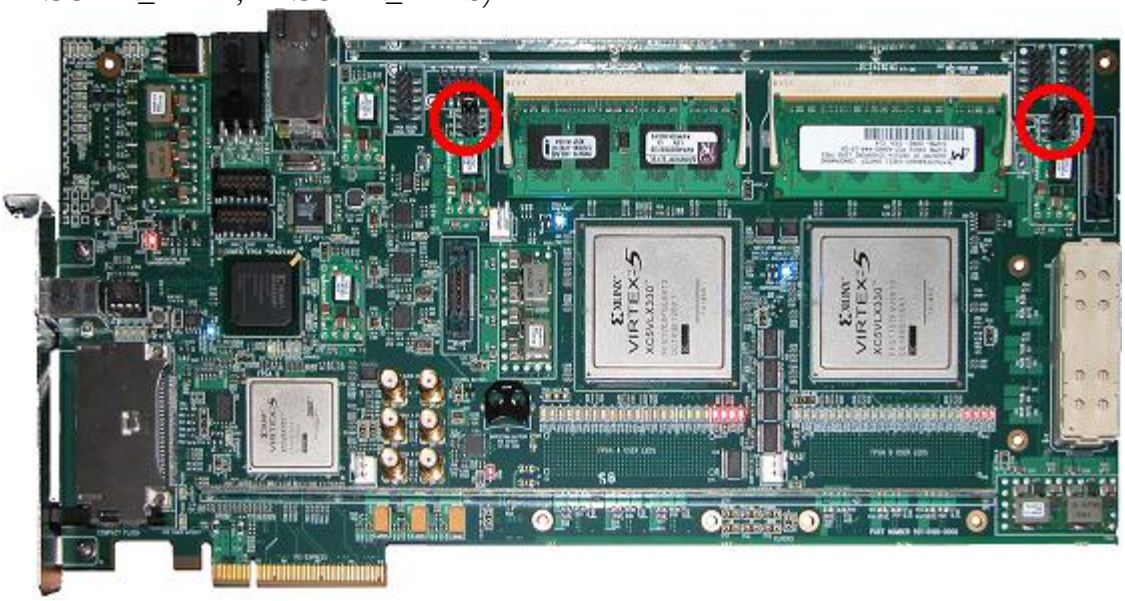


Figure 87 - DIMM Voltage locator

The jumper blocks for the two DIMMs are located next to the DIMM sockets. The one on the left controls DIMM A and the one on the right controls DIMM B.

18.1.3 DIMM warning LED

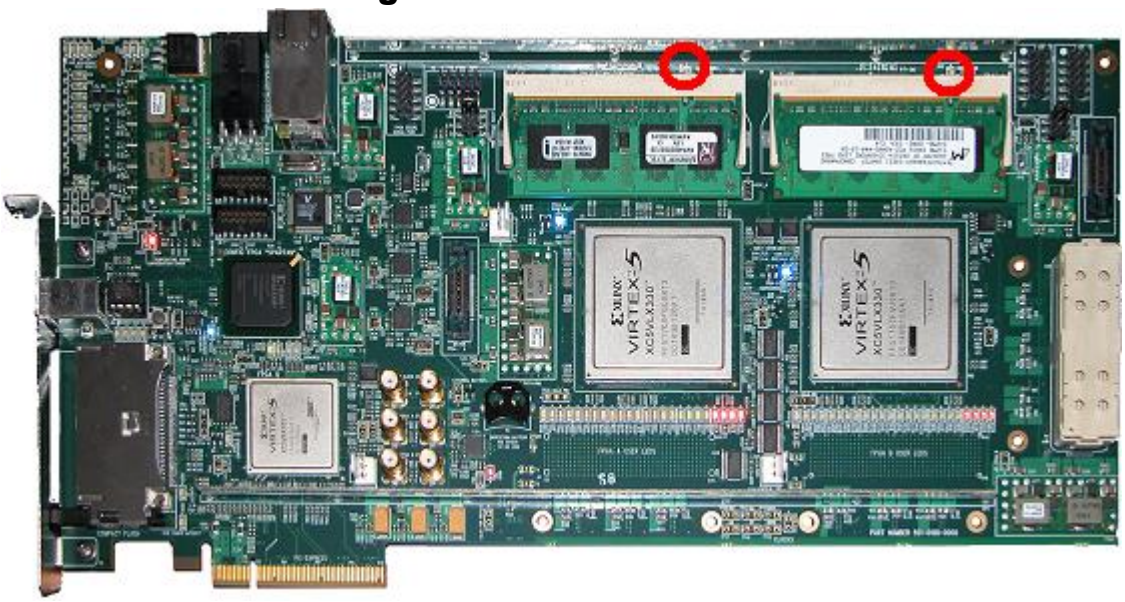


Figure 88 - DIMM warning LED locator

When the DIMM voltage is something other than 1.8V, there is a red LED that lights next to the DIMM. This LED means that you should get a voltage probe and measure the voltage being

supplied to the FPGA and DIMM. If this voltage is above 3.3V, you could be damaging your FPGA.

18.2 Clocking

The data signals in the DDR2 interface are clocked source-synchronously. In order to clock in and out the "DQ" data signals, the DQS signal is used as a clock using the Virtex-5 "BUFIO" clock driver. Details on how to implement a DDR2 controller are in the Xilinx application note XAPP858. You can also see the provided DDR2 reference design for example code.

A basic block diagram of the clocking is given below.

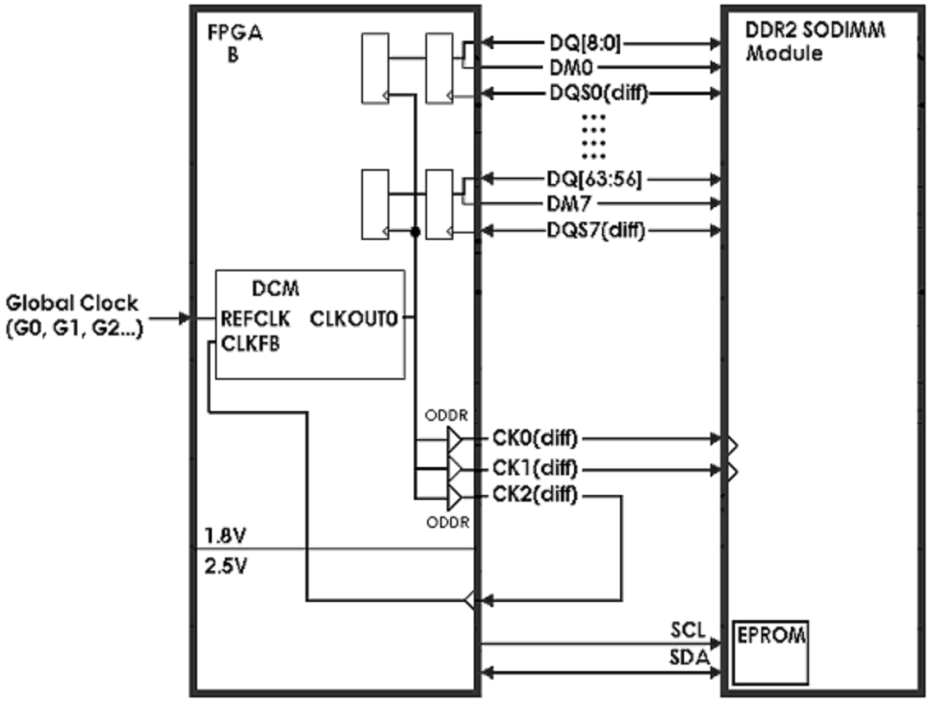


Figure 89 - DIMM clock diagram

Note that the DIMM_CK2 signal is driven by the FPGA from a 1.8V bank. The output should be a DIFF_SSTL18. It is received by a global clock ("GC") pin on the Virtex-5 device. To receive the signal, use an LVDS_EXT input with DIFF_TERM attribute set to TRUE.

The CK0, CK1 and CK2 signals are length-matched, so this input should be synchronous to the clock input of the DIMM module.

The DQ and DM signals are synchronous to the DQS signals in each bank. See the DDR2 SODIMM module specification for information on the timing of this interface.

18.2.1 DQS timing

In order to clock the DQ and DM inputs using the DQS signal, you can use a BUFIO clock buffer on the DQS signal. The provided DDR2 controller does not use this method. (It dynamically adjusts a DCM global clock for inputs)

18.2.2 Serial Interface

The SDA and SCL interfaces are connected to 2.5V LVCMOS buffers. External pull-ups are provided on these signals. The address of all DIMMs on the DN9200K10PCIE8T is set to zero. You can (optionally) read the IIC prom off the DDR2 SODIMM to dynamically determine the correct settings for the DDR2 controller. The provided DDR2 controller does this. Or, you can use our provided DDR2 controller to read the IIC contents of the DIMM, then use this information to configure your own DDR2 controller.

The SDA and SCL signals are also routed to GCLK signals on the FPGA (2.5V). These signals can be used as clock inputs on daughtercards of the SODIMM form factor.

18.2.3 Timing

The length matching of the DDR2 interface signals includes all signals except for DIMM_SCL and DIMM_SDA signals.

Due to the source-synchronous clocking techniques used by the DDR2 interface, the delay from FPGA to DIMM should not be needed, but is provided here anyway.

DIMMA	0.658 ns
DIMMB	0.623 ns

The trace impedance to each of the connectors is controlled to 50Ω. All signals in the interface are ground-referenced. Note that this is contradictory to the recommendations of the DDR2 SODIMM specification.

To increase the setup time available for control signals, modules may be set into T2 mode. In the reference design, the modules are in T1 mode.

Address and Control signals:

FPGA:

Assume a DCM in system-synchronous mode.
Worst clock-to-out time of Virtex 5 : 3.37 with DCM. No phase-shift.
Worst setup time: 0.097
Worst hold time: 0.21

DIMM:

setup 600ps
hold 600ps

DQ signals:

DIMM:
DQS must be within 350ps of DQ, DM
setup 400ps
Hold 400ps

FPGA:
IDELAY
setup –1.23
hold 2.14
clock-to-out 5.34

18.3 Compatible Modules

The list is in a later chapter. (Ordering information)

18.4 Incompatible Modules

Figure 90 - Lunar Module

18.5 Test points

Each DDR2 interface exposes five signals as test points, located on the bottom of the PCB right under the SODIMM connector. These signals are DQ0, DQS0p, CK0p, RAS# and CAS#. The test points are labeled in silkscreen. The test points near DIMMA implicitly are part of the DIMMA interface, and so on.

19FPGA Interconnect.

The point-to-point interconnect on the DN9200K10PCIE8T is designed to operate at the maximum switching frequency possible on the DN9200K10PCIE8T. The fastest switching standard available on the Virtex 5 FPGA is LVDS. Using this standard on interconnect of a DN9200K10PCIE8T; we have demonstrated switching frequencies as high as 950Mbs.

A block diagram of the point-to-point interconnect is below.

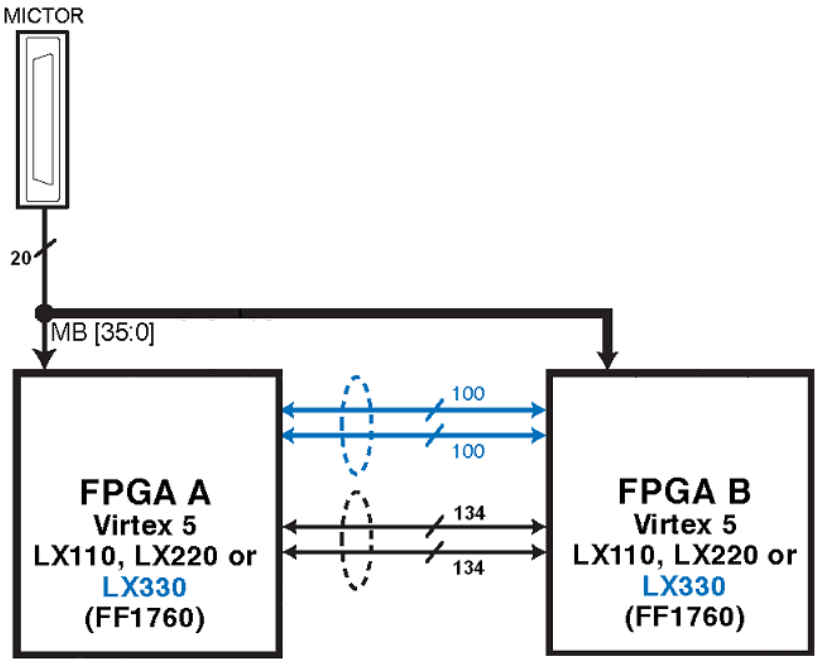


Figure 91 - Interconnect block diagram

The interconnect in the above diagram is confusingly described as sets of two busses. "AB" is the bus between FPGA A and FPGA B. It contains:
100 "p" signals that are available only if you have two LX330s.
100 "n" signals that are available only if you have two LX330s.
134 "p" signals that are always available.
134 "n" signals that are always available.

This is a total of 468 signals that can be used between A and B (that don't also have another purpose).

Each FPGA-to-FPGA interconnect signal is tested at 900 Mbs prior to shipping, no matter which speed grade is installed on your board. Higher speeds are possible, given appropriate IO timing methodology and speed grade parts.

Virtex-5 parts are advertized to go as fast as 1.2 Gbs, but I haven't tried it (The Dini Group reference design implements an older method from a Virtex-4 app note). Information on how to achieve this interconnect switching speed can be obtained by examining the Xilinx application note XAPP855. Other methods of implanting high-bandwidth interconnect are described in XAPP860.

In a synchronous system between two FPGAs and a DCM in zero-delay mode, the following timing is possible.

Clock-to-out	3.4ns
Trace Delay	1.7ns
Clock Skew	0.2ns
Duty Cycle	0.05ns (DDR mode only)
Jitter	0.1ns (adjust for BER)
Setup	1.0ns
	6.4ns

Maximum Frequency: 156 MHz

If LVDS is used, make sure to assign the DIFF_TERM attribute to the IBUFDS in the receiver FPGA.

As the frequency of synchronous communication between FPGAs increases, the user must implement more difficult techniques. Some of these techniques are described below, with a rough frequency range for their implementation.

0	MHz	Whatever
20	MHz	The user should use the "Pack the IOBs" by using synthesis attributes. The output delay for each output and setup time for each input is a known value.
100	MHz	Use DCMs in each FPGA to eliminate the variation of clock network skew internal to each FPGA and to reduce clock-to-out time.. The clock must be free-running
250	MHz	Use DDR clocking, and DDR IO buffers
300	MHz	Use source-synchronous clocking between FPGAs. The clock is driven with the data for each bus. The receiving FPGA uses the clock signal, received on a "CC" pin to clock the IOs in the bus. An IDELAY element on the CC pin input delays the clock with respect to the data by a fixed amount to allow some setup time.
550	MHz	Use the Virtex 5 build in ISERDES and OSERDES modules.
600	MHz	Use Virtex 5 PLL devices to reduce cycle-to-cycle jitter on the clocks.
700	MHz	Individually de-skew each bit using IDELAY elements. Use a training pattern or hard-code the correct delay values for each input.
800	MHz	Use LVDS signal standard
900	MHz	Dynamically de-skew each bit to account for temperature and voltage variation
1+	GHz	Highest speed grade parts are required.

Note that for speeds above 550 MHz, you must use the ISERDES and OSERDES modules, which add latency to your interconnect. (At speeds greater than 500 MHz, there is more than one clock-cycle of latency in board trace delay alone).

Also note that when using either the ISERDES or IDELAY technique, the latency is no longer fixed between the FPGAs, and per-lane "cycle" de-skew will also be required.

For the maximum bandwidth between two parts, use single-ended signaling at 700 MHz. For single-ended signaling, an IOSTANDARD of LVCMOS25 is appropriate. Use drive strength of 6mA or 8mA. When using single-ended signaling, the SSO limits of the device must be maintained. You could do this by having multiple output phases, by balancing the number of outputs and inputs on a single bank, or by applying a switching-balanced parallel encoding to the data.

20Main Bus

Main Bus is the interface that the DN9200K10PCIE8T uses to bring USB and PCIe access to both of the Virtex-5 FPGAs. If you want to use USB in your design, or want PCIe access without implementing PCIe in FPGA, then you must implement a Main Bus slave in your FPGAs. The reference designs include one such controller, and you are free to use it.

Drive strength.
Please use the highest drive strength IOs available (24mA)

20.1 MB Signals

The DN9200K10PCIE8T, in addition to the dense interconnect available between FPGAs in a point-to-point topology, provides a 36-signal-wide "MB" bus that is connected to both Virtex-5 FPGAs.

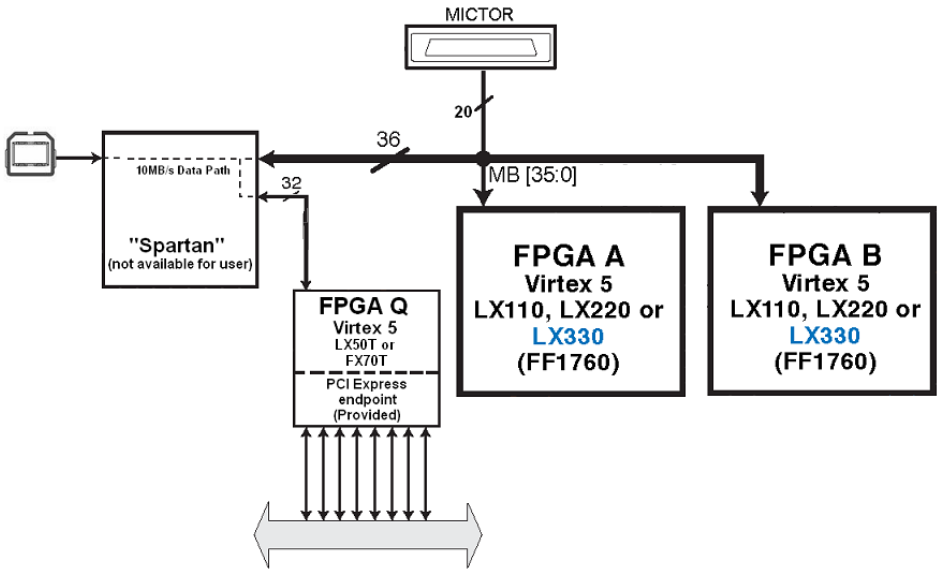


Figure 92 - Main Bus block diagram

These signals are reserved for USB and PCI Express communication using the "Main Bus" interface.

20.1.1 MB vs. MainBus Disambiguation

I try my best to say "MainBus" when I am talking about the interface definition that allows FPGAs to access USB and PCI Express. I try to say "MB" when I'm talking about the actual 36 physical signals that these interfaces use.

20.1.2 Electrical

The MB signals are fixed at a 2.5V signaling level. LVCMOS25 is an appropriate singling standard. Due to heavy capacitive loads on the MB signals, you should use drive strength of 24mA to use main bus. DCI should not be used because the signals are not impedance-controlled. Although not required, by convention, data on the MB signals is synchronous to the MB48 clock. In order to use the "Main Bus" interface to communicate with USB or PCI Express, you must use the MB48 clock. This clock runs at a fixed 48 MHz.

Note that as well as the 36 "MB" signals, there are also 16 signals in the "selectmap_d[15:0]" that connect to all FPGAs that could be used for user data. Dini Group does not directly support using these signals. If you chose to use these signals, note that the FPGA design can interfere with the programming of FPGAs. You would have to keep the outputs on these signals tri-stated until all FPGA configurations are complete.

20.1.3 Timing

As described above, the MB signals are typically run synchronous to the 48 MHz CLK_MB48 clock. The delay for each main bus trace is not given. However the interface is at least fast enough to run synchronously at 48 MHz.

You may be able to achieve performance from FPGA-to-FPGA on this bus as high as 125 MHz, or higher if you adjust input and output clocks and perform a timing analysis.

20.2 Error Codes

The Main Bus interface has no way of signaling an error condition on read requests, but some errors will result in the same sentinel values being returned. Following is a list of these values.

0xABCDABCD: The Main Bus read timed out. (PCIe only)

0xDEADDEAD: The Main Bus read times out (USB only). When this condition occurs, a register, accessible as part of the "configuration register" space, increments. In this way, it is possible for a Main Bus access program to verify that a MainBus transaction has succeeded.

0xFFFFFFFF: The PCIe bus timed out. This is not a value returned by the DN9200K10PCIE8T. The PCIe request was not returned. FPGA Q may not be configured correctly.

0xDEAD5566: This value is returned by the Dini Group reference design as a default value, when a read request is to an address that has no registers associated with it.

0x12345678: The Main Bus is disabled. This is the default state of the DN9200K10PCIE8T when it powers on. To set the DN9200K10PCIE8T to enable, a configuration register must be written. This behavior is intended to protect users who do not wish to implement Main Bus interface, but who wish to use the MB0-MB35 signals for their own purposes.

20.3 Main Bus FPGA Interface

All memory-mapped transactions in the reference design occur over the MB bus. This 36-signal bus connects to all Virtex 5 FPGAs and to the Spartan 3 configuration FPGA. The Configuration circuit (Spartan 3) is the master of the bus. All access to the MB bus (reads and writes) is initiated by the Spartan 3 FPGA when the reference design is in use.

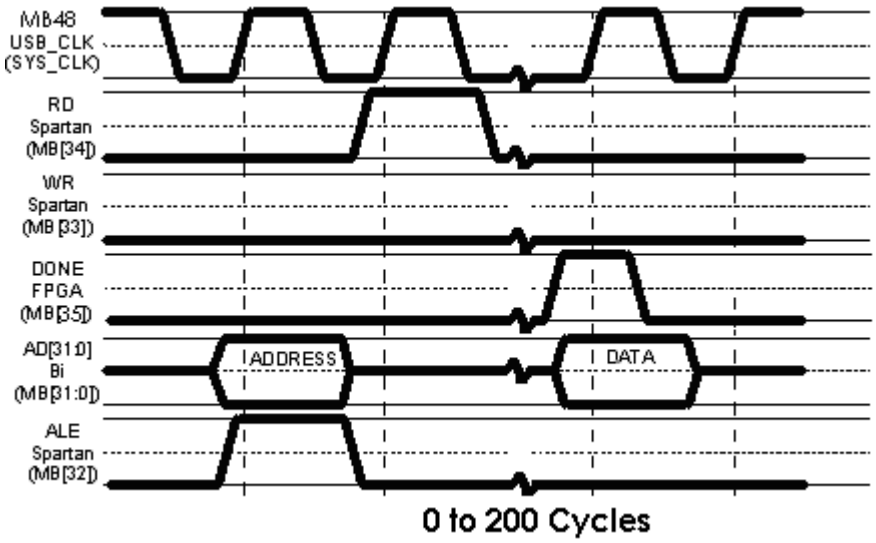


Figure 93 - Inaccurate Main Bus read timing

All transfers a synchronous to the CLK_MB48 signal. This clock is fixed at 48 MHz, and cannot be changed by the user. This clock is LVCMOS, single-ended. For best performance, the highest available drive strength in the FPGA can be use. When the configuration circuit asserts the ALE signal, the slave device on the bus (the FPGA) is required to register the data on the on AD bus. This is the "main bus address". All future transfers over the main bus are said to be at this address, until a new address is latched. On a later clock cycle, the master may assert the "RD" signal. Sometime after this, (within 200 clock cycles), the FPGA should assert MB_DONE for one clock cycle. On this cycle, the master (Spartan) will register the data on the AD bus, and that will be the read data. If MB_DONE is not asserted, then a timeout will be recorded and the transaction cancelled.

Here is a write transaction:

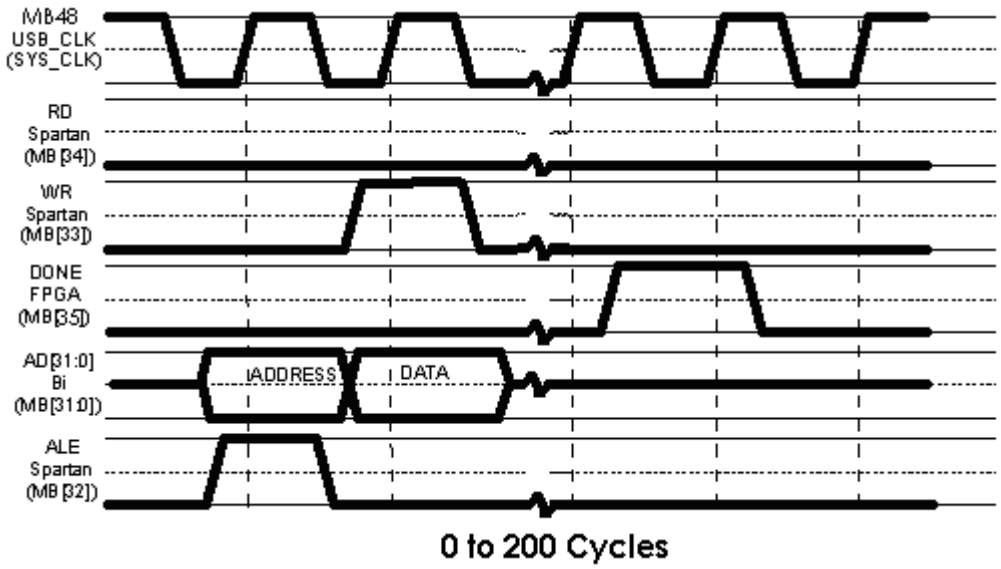


Figure 94 - Inaccurate Main Bus write timing

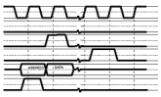
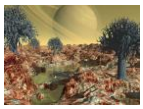
When the Spartan asserts the "WR" signal, the FPGA should register the data on the AD bus.

Sometime after this, the FPGA should assert the MB_DONE signal. This will allow the Spartan to begin more transactions. The FPGA may delay this for up to 200 clock cycles before a timeout is recorded and the transaction is cancelled.

Main bus can be controlled from the USB Controller program. (Read and write single addresses, or to/from files) It can also be written from the main.txt configuration method. The main.txt syntax is

MAIN BUS 0x<address> 0x<data>

Where <address> and <data> are 8-digit (32-bit) hexadecimal numbers.

Cycle Count:	1.0×10^{0}	1.0×10^{5}	1.0×10^{10}	1.0×10^{15}	1.0×10^{20}	1.0×10^{25}
Behavior:						

20.3.1 mb_target.v

A file is provided that can be used as a drop-in MainBus target interface. It also implements the conventional memory allocation between FPGAs by the use of a compile-time parameter. In order to change the conventional memory allocation, you will have to modify mb_target.v

20.3.2 Conventional Memory map

By convention, FPGAs on the main bus interface are assigned address ranges. Assigning address ranges is required because the "FPGA sourced" signals (MB_DONE) need to be driven by only one FPGA at a time.

The convention that Dini Group uses is to reserve the upper four bits in the address as an FPGA-select address. The address range (hex)
0x00000000 – 0x0FFFFFFF is reserved for FPGA A,.
0x10000000 – 0x1FFFFFFF is reserved for FPGA B,
and so on.

The user need not follow this convention, but unless you really need 32-bit addresses, we recommend using it. Only one FPGA has "control" of the DONE signal. If the last address latched by ALE was not for a given FPGA, it should tri-state the output. Before tri-stating any signal with a pull-up or pull-down resistor, it is good practice to drive the signal to the DC value before tri-stating. (So that simulation will match emulation result).

21Ethernet

An Ethernet interface is available to FPGA A. It is provided by a Vitesse VSC8601 tri-mode Ethernet PHY. The RJ45 connector can be used to connect to a regular 10Base-T, 100Base-TX, or 1000Base-T Ethernet network connection.

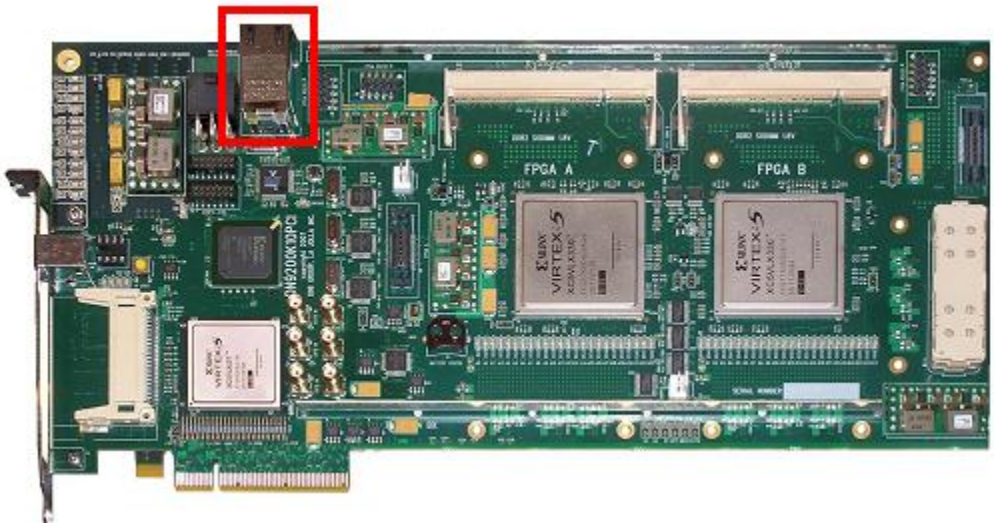


Figure 95 - Ethernet locator

The VCS8601 device does not contain an Ethernet MAC. The FPGA must implement a complete network stack to make use of the Ethernet connection.

http://www.opencores.org/projects.cgi/web/ethernet_tri_mode/overview

21.1 RGMII

The 4-bit GMII interface is the only strictly required interface on the PHY device. The EEPROM, MDIO, and other signals are only required if you want to put the PHY into a mode that is not default.

The SMI (MDC, MDIO signals) address is set to 0000.

21.1.1 Electrical

The appropriate electrical standard to use is LVDCI_25. In Gigabit mode (default), the MII interface runs at 125MHz, DDR.

The CLK_ETH125 signal should use the SSTL_II_25_DCI signaling standard.

21.1.2 Timing

The board is designed intending for a particular use model for the IO timing.

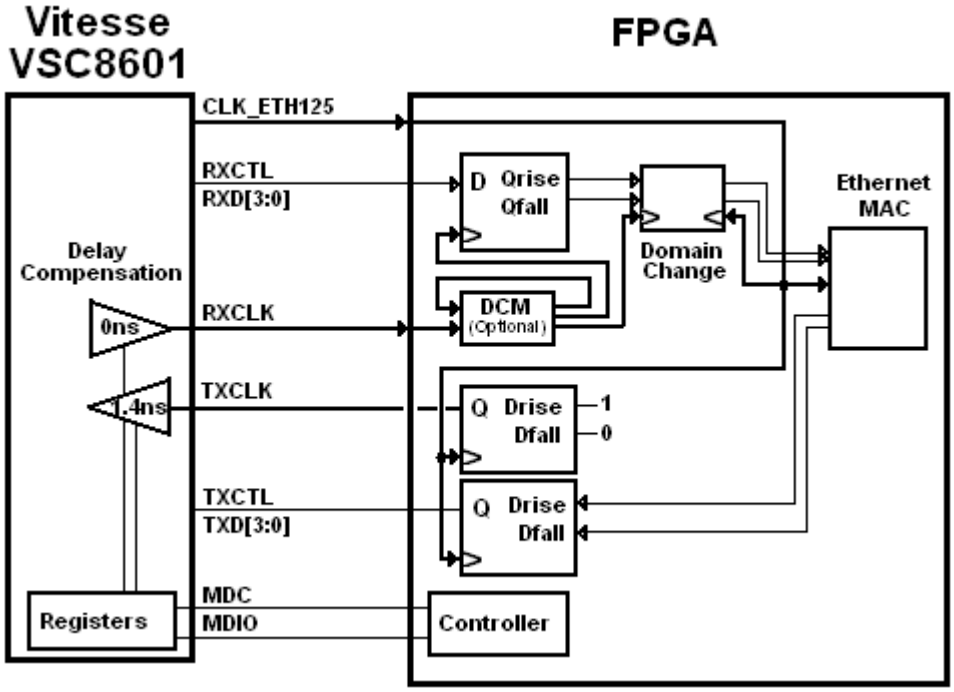


Figure 96 - Ethernet timing

The clocking plan here assumes you are running in gigabit mode. If in 100 or 10 megabit mode, then some other thing might be required.

The interface requires a 125 MHz system clock. The part conveniently provides this with the CLK_ETH125 signal. This signal should be used to drive the TX interface and the MAC controller.

For the TX interface timing, you can output clock and data with zero skew between them, as shown in the above diagram, and set the TX clock compensation register in the Vitesse part to meet the setup and hold time requirements. Alternately, you can do something else. In order to output a clock with zero skew from the data, you use a output DDR register (ODDR) with the rising edge data set to 1 and the falling edge set to 0.

For input timing, you can clock the RX data signals off the RXCLK, and then make an asynchronous domain change to the Ethernet MAC, or you can figure out what the correct

phase offset is between the CLK_ETH125 signal and the RXCLK signal and make a synchronous domain change. We do the former because RGMII is very easy to put into an elastic buffer.

All signals on RGMII are skew-matched on the board to within: 100ps

FPGA:
DCM is in system-synchronous mode with no phase adjustment

Worst clock-to-out	3.37
Worst setup time	0.097
Worst hold time	0.21

PHY: (clock measured at PHY pin)

clock-out	2ns
setup	2ns
valid	1.2ns

21.2 Configuration Registers

In order to read and write registers on the Vitesse part, you must implement a MDIO controller. You will probably need to look at the Vitesse datasheet to see where register locations are and IO timing, etc. This step is probably required, because the default register settings may or may not be what you want.

If you do not implement the MDIO interface, then the default settings are used for the device. This includes settings that are specified by multi-level inputs connected to resistors.

The CMODE options of the Ethernet PHYs has been set as follows

CMODE0 – 0100 (8.25 kΩ resistor)
CMODE1 – 0000 (0 Ω resistor)
CMODE2 – 0001 (2.2 kΩ resistor)
CMODE3 – 0000 (0 Ω resistor)

This results in the following settings

ADDR	=	00000	MDIO address
CLKOUT	=	TRUE	Drives the CLK_ETH_125 signal
PAUSE	=	00	I don't know
DOWNSHIFT	=	FALSE	I don't know
SPEED	=	00	Gigabit mode only
ACTIPHY®	=	FALSE	I don't know what this is.
SKEW	=	11	This controls the MII timing. It probably won't work until you set this.
MAC CALIBRATION MODE	=	00	????????

The LEDs on the RJ45 connector are controlled by the PHY. The Amber LED indicates activity and the Green LED indicates link in gigabit. The LED, DS64, located next to the RJ45 connector, indicates link in 100Mbit mode. The 10Mb link LED is not configured.

Hot plug is acceptable on a 1000Base-T connection.

The Ethernet PHY works with the Xilinx Ethernet IP, but only in 10 and 100Mbit modes.

21.3 MII Interface

The physical interface is 1000Base-T, 100Base-T or 10Base-T. It has an "RJ45" style modular connector. It is connected through a transformer. It is hot-swappable.

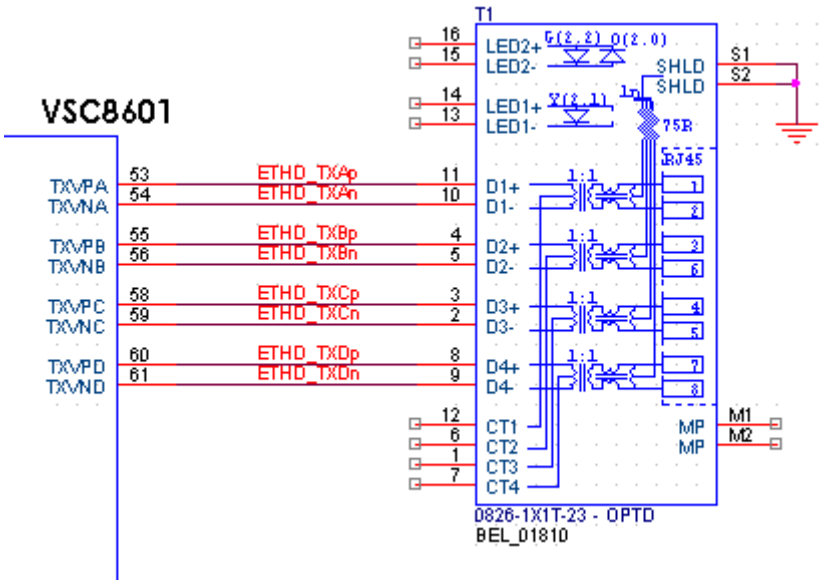


Figure 97 - 1000Base-T circuit

The above schematic clipping is useless but looks cool and technological.

I don't know what else to say about this. Look up 1000Base-T

21.4 External EPROM

Every FPGA that has an Ethernet connector on it also has a very small EPROM. This is typically used to store a MAC address and phone numbers. The limited details about it are in another section.

21.5 EPROM PHY Configuration

The EEDAT and EECLK signals are intended to connect the PHY to an EPROM that would contain configuration settings for the device (LED behavior, MII timing, Link speed, duplex, auto negotiation, etc.). Since the MDIO interface is connected to the FPGA, it is unlikely you

would ever use these signals, unless you just like emulating EPROMs on weekends and vacations.

This can be used instead of the MDIO interface.

21.6 JTAG

The VSC8601 device is attached to a JTAG chain. I don't know why you would need access to this. It isn't tested or thought about ever. This JTAG chain does not connect to the FPGA JTAG chain. It's 3.3V.

21.7 Ethernet MAC

There is no MAC provided. You might think "I can use the Virtex-5 built-in tri-mode MAC!!" However, you'll be disappointed because this isn't available in the LX330. You can route the MII interface all the way over to the LXT (FPGA Q) and use its hard MAC if you want. This wouldn't be very hard.

You can also buy access to the Xilinx soft MAC. You probably need to implement a processor and a software network stack. The way we did it is using the Xilinx demonstration version of their 10/100 MAC, and connected it to a Microblaze running lwip stack. We had to write a converted between GMII and RGMII, which is basically just adding a DDR flip-flop.

22EPROM

A small EPROM (1 kΩ) is attached to FPGA A. These devices are intended to store identification data for generating a unique MAC address for the Ethernet interfaces. However, the EPROM can be used for any user-defined purpose requiring static-memory intensive tasks, like remembering your name and birthday.

The interface to the EPROM is a standard IIC at 1.8V. The IIC address of the devices is (binary) 1010 000

The maximum clock speed of the IIC interface is 400 kHz

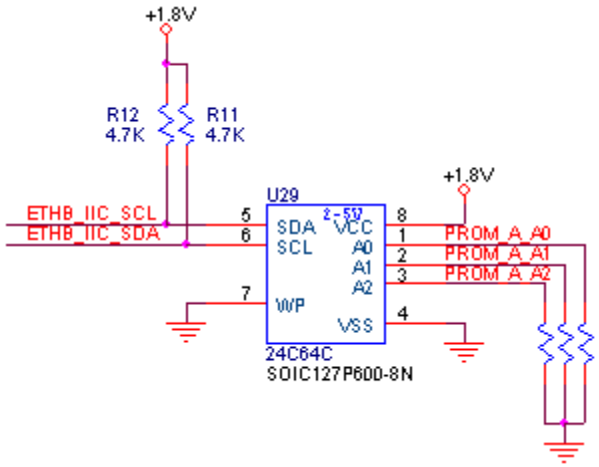


Figure 98 - EPROM circuit

23SPI Flash

For non-volatile memory needs, a medium-density SPI serial flash is provided on each FPGA.

23.1 On FPGAs A and B

The SPI flashed on FPGA A and B are 16 Mb, part number AT45DB161D. You should look in the datasheet for this part to see the IO interface and timing requirements. The signals are LVCMOS25. The flash devices cannot be used for configuration, only for user data.

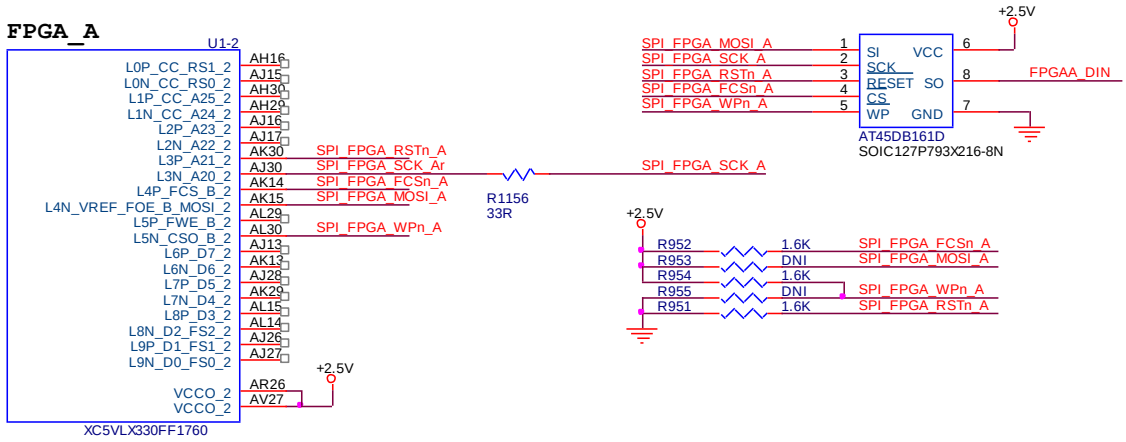


Figure 99 - SPI Flash circuit

Please note that the input signal "DIN" connects to the "DIN" pin of the FPGA. This pin cannot be placed like a normal IO. In order to access this pin as an input, you need to instantiate a STARTUP_VIRTEX5 in your design, and use the DINSPI port of that module. Also, since nobody knows the timing of that port, we have no idea what the maximum speed of the SPI interface is.

23.2 On FPGA Q

One FPGA Q, the situations is similar, but with some important differences. The signal standard is LVCMOS33. The part number is AT45DB642D. The part can and should be used for configuring the FPGA. However, if you are very clever, you can also use the flash for user data. In the same way, the DIN input needs to be gotten from the STARTUP_VIRTEX5 module. Additionally, the SCK signal needs to be driven from the USRCCLKO port of the STARTUP_VIRTEX5 module.

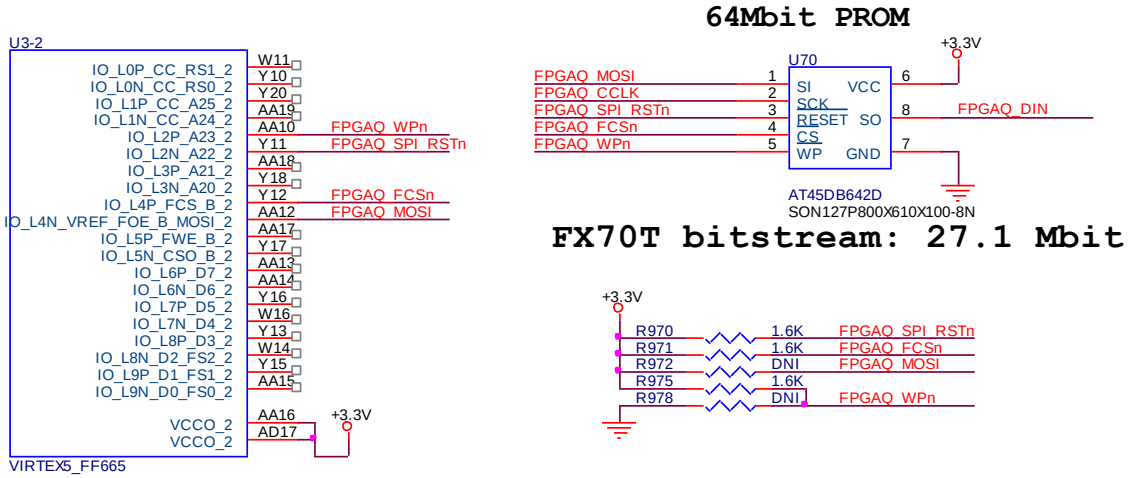


Figure 100 - SPI Flash circuit Q

In order to program this flash with a bit file, you can use the Xilinx program iMPACT. From here you can select the FPGA Q (last item on the JTAG chain) and chose "program SPI flash". The iMPACT program will automatically load the FPGA with a bit file that allows the programming of the flash, program the flash using that bitfile, then program the FPGA with the bit file that you just loaded into the flash using JTAG. See the section on "updating firmware", as that section has helpful things like screen captures and proofreading.

24Mictor Connectors

There are three 38-pin "Mictor" connectors on the board for the purpose of using a logic analyzer. (If you are still using a logic analyzer- they are *so* 2002) Consider using an embedded logic analyzer instead like ChipScope ($500). This logic analyzer places-and-route within your design, either in the RTL, or post-synthesis. They are more flexible than a stand-alone analyzer and can simultaneously access more signals and triggers.

Although the Mictors are designed to be used with a logic analyzer, they can also be used for cabling two boards together, or to a daughter card, or just for use as test points. The "trigger" signals connect to clock-capable IO pins, and so can be used as low-skew clock inputs.

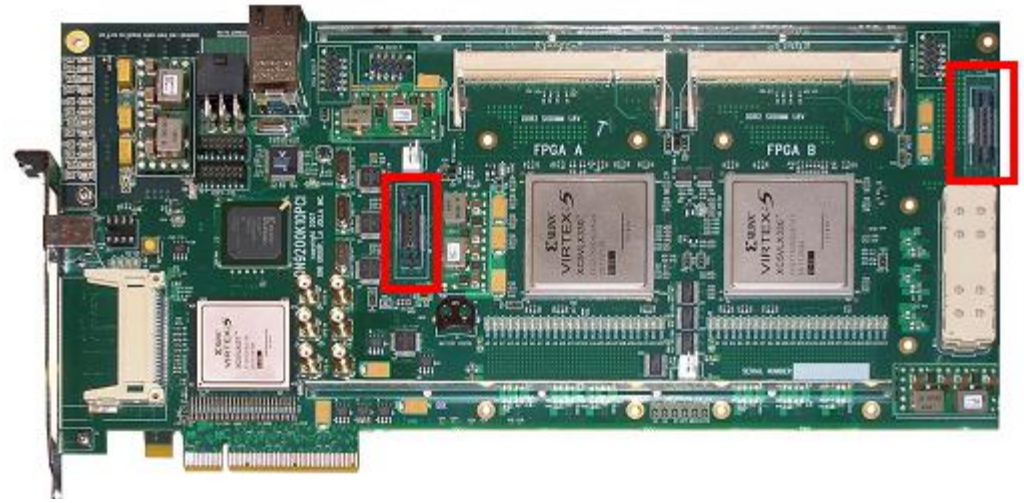


Figure 101 - Mictor locator

Hot-plugging a Mictor connector is generally safe. When connected to a logic analyzer, signals MICTOR32 and MICTOR33 can be used as trigger signals. I've never actually used a logic analyzer; I have no clue what I'm talking about.

Figure 102 - Mictor cable

Signals connected to the Mictor are 50Ω. DCI and SSTL (referenced input) can be used on the Mictor interface.

24.1 FPGA A Mictor

The Mictor connected to FPGA A has a total of 34 signals (32 plus two triggers). The voltage level of each signal is determined by the voltage level of the bank that the signal connects to. You may need to change the trigger level of your logic analyzer. The "daughter card" voltage banks (when no daughtercard is installed) are 1.2V (use a 0.7V reference level).

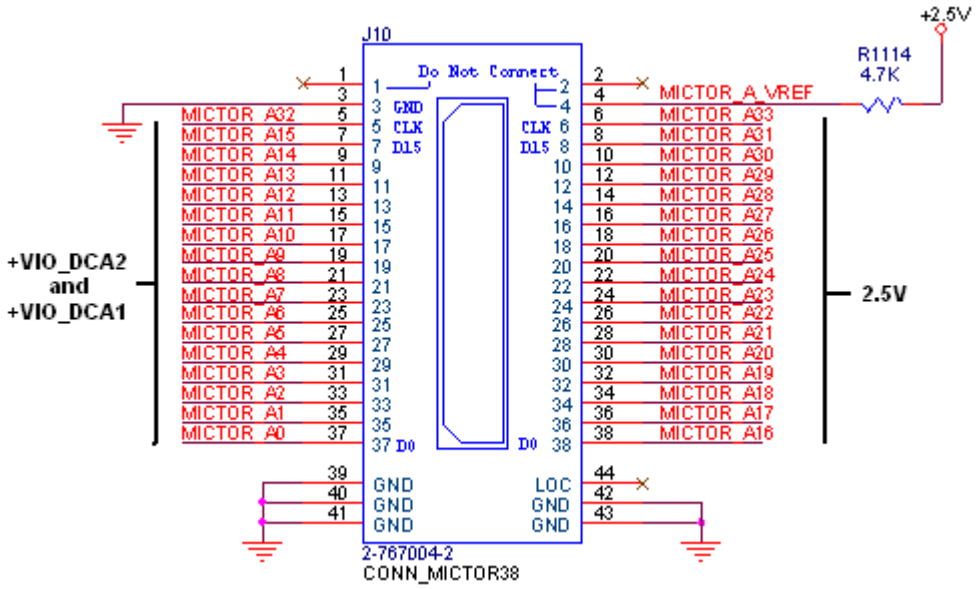


Figure 103 - Mictor A circuit

This diagram shows how the voltages are controlled on the Mictor connector. The +VIO_DC* voltages can easily be changed if needed.

24.2 FPGA B Mictor

The FPGA B Mictor is pinned out exactly like the one on FPGA A, but the voltage splits are different. The daughtercard bank voltages are 1.2V (use a 0.7V reference). This voltage can be changed easily if needed.

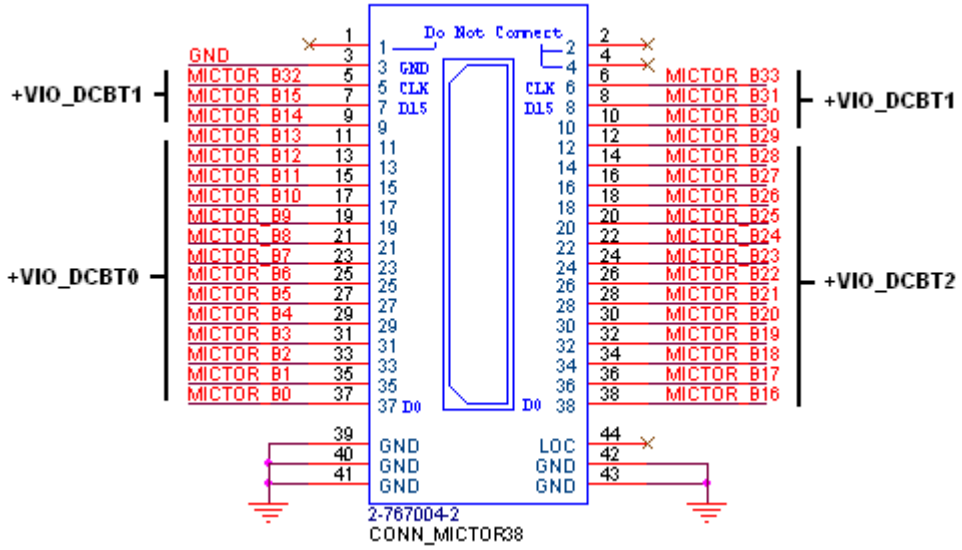


Figure 104 - Mictor B circuit

24.3 MainBus Mictor

A second Mictor connector, on the backside of the board, is connected to the MainBus and SelectMap interfaces of the DN9200K10PCIE8T.

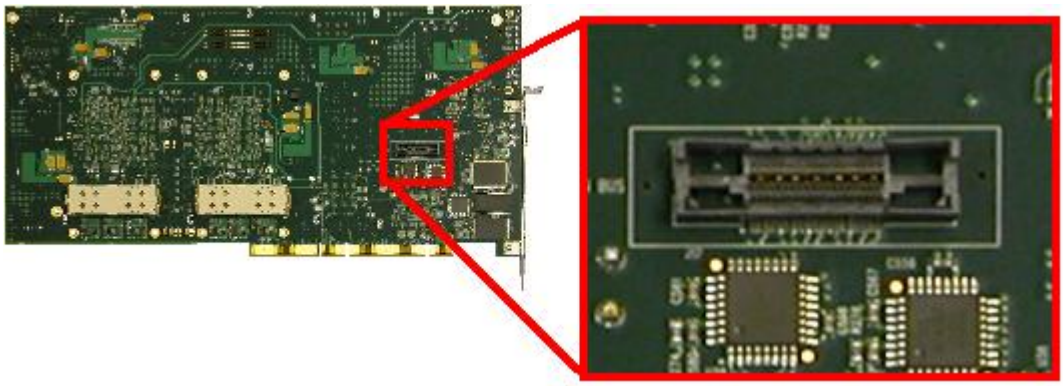


Figure 105 - MainBus Mictor locator

Most of the signals attached to the Mictor are accessible from both FPGAs on the DN9200K10PCIE8T. Since these signals are heavily loaded, this connector is less suitable for high-speed signaling.

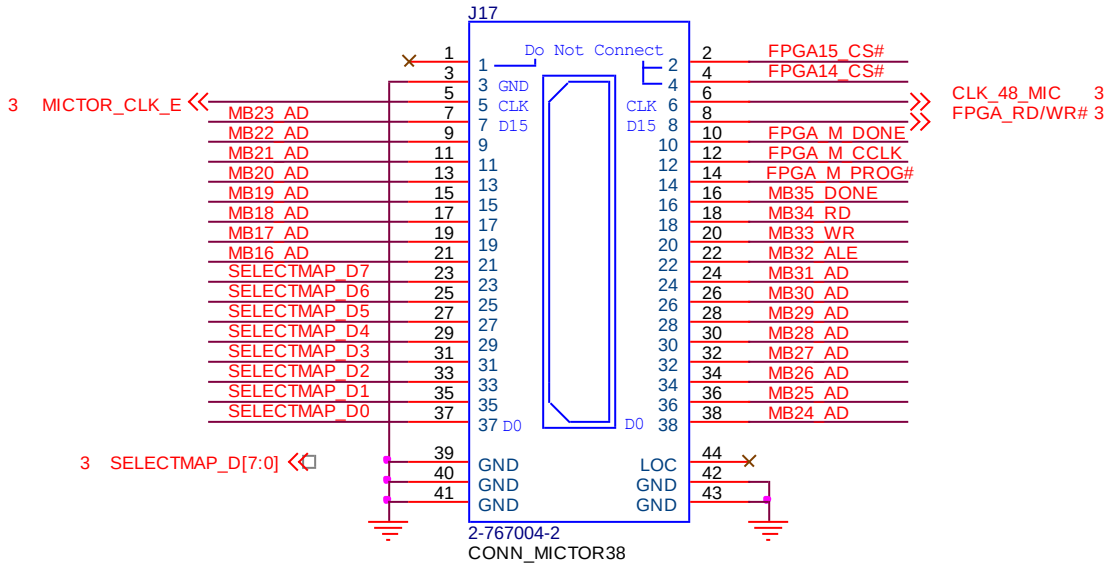


Figure 106 - Main Bus Mictor circuit

The "clock" or "trigger" signals on this connector, CLK_48_MIC and MICTOR_CLK_E are driven at a fixed 48 MHz. If you need to use a logic analyzer, this is the only available trigger.

All signals are 2.5V (use a 1.25V reference).

If you use the signals SELECTMAP_D[7:0] for any purpose other than configuration, care must be taken to prevent the FPGAs from driving these signals before all FPGAs are configured, or else risk interfering with the configuration process.

Some SelectMap control signals are connected to this connector, but are not user-accessible.

This connector could potentially be used for configuring Virtex FPGAs on daughtercards. You would have to contact us for information about that possibility.

25Power

The power used by the DN9200K10PCIE8T is derived from an external 12V voltage supply. The current at these voltages is supplied through the PCI Express power connector, J3.

NO power is taken from the PCIe edge connector. Therefore, if installed in a PCI Express slot with no power connector, the board will not power on.

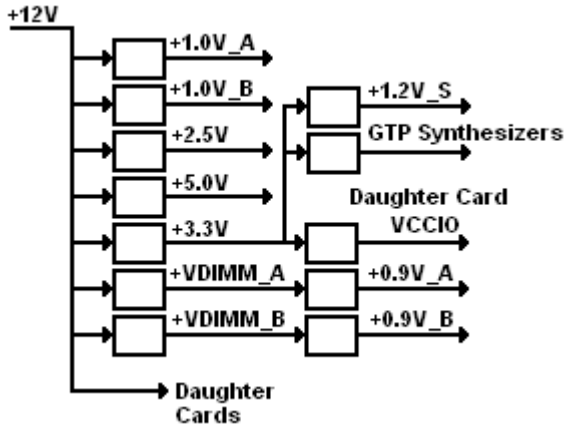


Figure 107 - Board power topology diagram

The maximum power draws on each of these rails is given below.

+12V	9A
+1.0VA	15A
+1.0VB	15A
+2.5V	20A
+3.3V	6A
+5.0V	9A
+VDIMM_A	2A
+VDIMM_B	2A
+1.2V_S	0.2A
+0.9VA	0.2A
+0.9VB	0.2A

25.1 Power 12V

The 12V rail is used to generate most other voltages on the board. The only places where 12V is used directly are the daughtercards.

Below is a list of the maximum power draw of each of the 12V loads on the DN9200K10PCIE8T.

Rail	Max Current	Uses	12V current
1.0V_A	25	Internal FPGA power	2.3A
1.0V_B	25	Internal FPGA power	2.3A
1.8V	2.5	DIMM B	0.3A
	2.5	DIMM A	0.3A
2.5V	9	Spartan 3 (1.2V) FPGA IO FPGA Aux power	2.6A
Daughtercards	10W		1.2A
TOTAL			9.0A

The total possible power requirement of the DN9200K10PCIE8T is 9A on 12V (108W).

More typically, each FPGA would only use 10W, and daughtercards would use little power (2W). Under these conditions, the 12V power requirement is only 2.5A (25W). Under these conditions use in a server rack would work.

25.2 Power 3.3V

3.3V is used by the DN9200K10PCIE8T to supply the clock distribution network, the configuration logic (Micro controller and Spartan 3 FPGA), and daughtercard power.

The maximum power requirement for the DN9200K10PCIE8T on 3.3V is 1A. Current for 3.3V is NOT taken directly from the ATX power supply or from the PCIe slot.

25.3 Power 2.5V

2.5V power is generated from the 12V using a 30A power supply.

25.4 Ground

All ground (0V) voltages on the DN9200K10PCIE8T are shared. A monolithic ground design strategy was used. The nets GND_SHIELD and GND_ANALOG are directly connected to the ground plane.

25.5 Voltage Regulation

Within 2% typically

25.6 Power Connections

The primary sources of power for the DN9200K10PCIE8T are the PCI Express "graphics" power connector. From these two sources, the DN9200K10PCIE8T draws current at 12V; all other voltages on the board are generated.

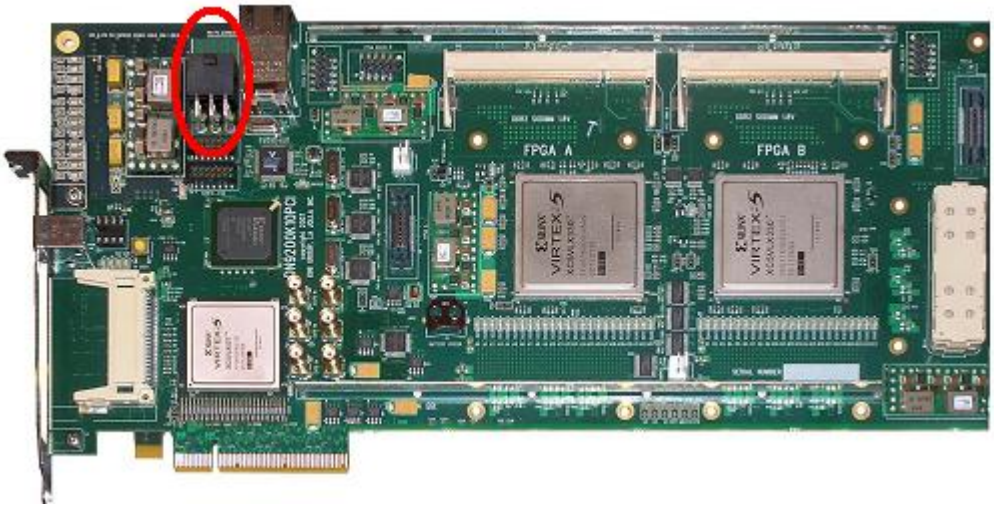


Figure 108 - PCI Express graphics power locator

This connector will work with a standard ATX power supply. Any supply rated above 300W is likely to be suitable for use with the DN9200K10PCIE8T.

If no 6-pin PCI Express "graphics power" connector is available, you may use an adapter cable (provided). Most new power supplies now have this connector available.

Note that only a 6-pin "PCI Express graphics" cable should be used. This is easily confused with the now-defunct "AUX POWER" connector (also 6-pin) and the 4-and 6-pin EPS "server motherboard" connections. The connector is keyed, so the wrong connectors will have difficulty fitting properly into the board.

Fittings are supplied such that the board can be powered from the PCI Express slot if this feature is desired; however this operation is not recommended because it can easily overload the motherboard.

25.7 Power Monitors

The DN9200K10PCIE8T monitors the voltage levels on the board to ensure they are within tolerance. If they fall out of tolerance (above or below voltage) the board will enter a reset state. These tolerance ranges are listed below.

1.0V	(0.95 to 1.21)
1.8V	(1.65 to 3.00)
2.5V	(2.20 to 2.90)
3.3V	(2.89 to 4.00)
5.0V	(3.99 to 6.02)

The following voltages are not monitored.
1.2V_S, VCCO_B0, VCCO_B1, VCCO_B2, DIMM_VTT, DIMM_VREF

When a power supply voltage falls out of tolerance, the board is put in reset (the SYS_RST# signal is asserted), and SYS_RSTn LED glows, and an LED along the right hand side of the board will light to indicate which power rail has failed.

The voltage levels are measured with a RC filter "time constant" of around 1 kHz. This means transient voltage spikes may not trigger a board reset.

25.8 Power Thru-hole Access points

Each power rail requiring more than 100mA on the DN9200K10PCIE8T has a dedicated test point associated with it. This test point is a through-hole, two-pin location, where pin one is the power rail, and pin two is a ground connection. These test point locations are suitable for supplying at least 2A, regardless of the power requirements or capabilities of the power net.

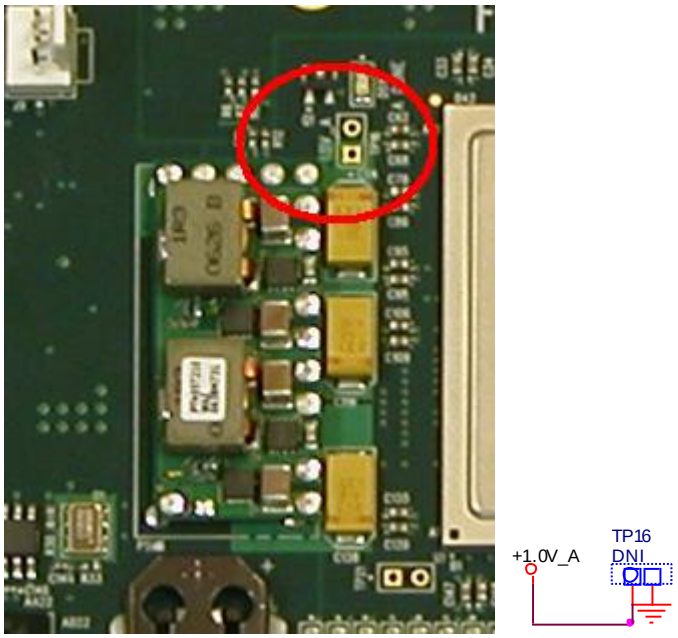


Figure 109 - Power Test points

Pin one is a square. Pin two is circular.

These test-points are suitable for wiring to if power is needed off-board for some reason. Maybe you need to bring power in from an external source.

25.9 Power measurement TP

The following test-points are located along the left edge of the board, next to an LED associated with that power net. These test points are square pads. They are not suitable for supplying power to the board, or off the board.

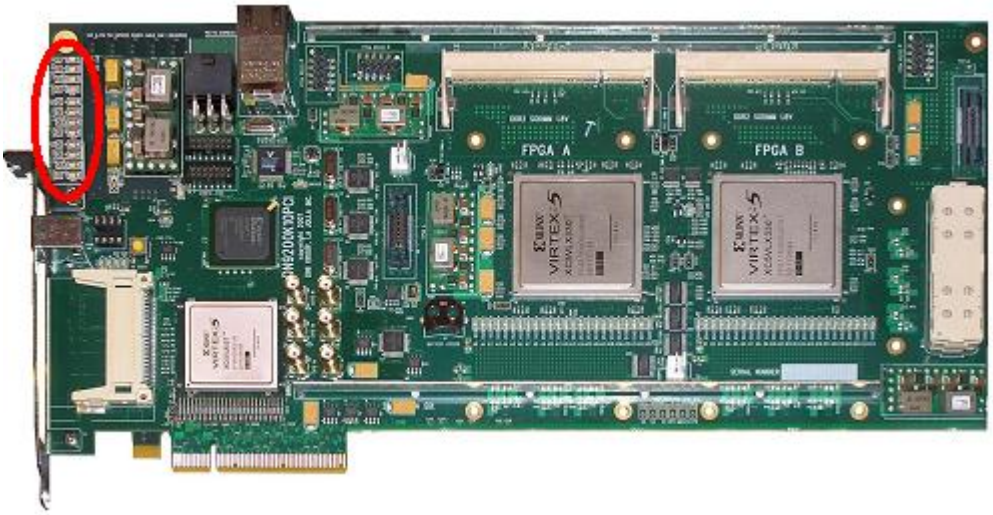


Figure 110 - Power Fail LED locator

Figure 111 - Power probe point circuit

The test point reference designator is not visible on the silkscreen of the DN9200K10PCIE8T. Instead, there is a label indicating which power net the test point is connected to. These test points are connected by thin traces that are not capable of conducting more than 100mA of current. You should only use these test points for probing. For noise measurements, it is better to use the test points next to each power supply.

25.10 Heat

The maximum power dissipation supported for each FPGA is 25W. Using the provided heat sink and fan assemblies, FPGAs will remain under the maximum recommended junction temperature (85°C). If your design exceeds this limit, you can assume the temperature of the device raises 2C° for each watt above this amount your design uses. Put this number in the settings of the timing analyzer.

Power requirements of a design can be estimated using the power estimator tool in ISE 10.1.

For this calculation the board is assumed to be in an ambient temperature of 35°C. In a closed computer case, the ambient temperature will increase.

We have alternate fans and Heatsinks that can help reduce the FPGA temperature. We can ship you some if you request.

25.10.1 Fans

The fan units attached above the heat sinks are powered by 5V. Each fan has its own power connector.

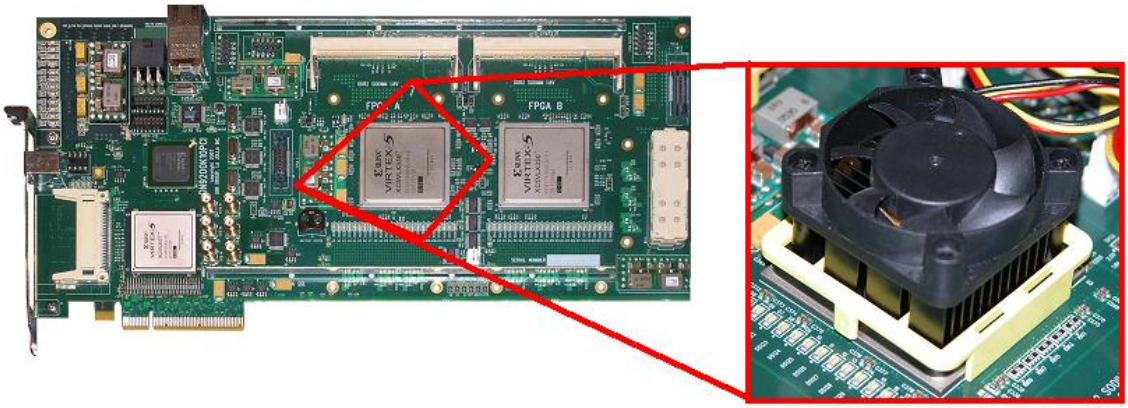


Figure 112 - Heatsink fan locator

The fans spin counter-clockwise in the northern hemisphere, or clockwise in the southern hemisphere.

25.10.2 Removing Heatsinks

The heat sink/fan assemblies are attached using a plastic clip. There is a thermal interface material between the FPGA and heat sink that is slightly adhesive. The easiest way to get them off is to unplug all the fan power and turn the board on. After a few minutes, turn the board off and then try to unseat the heat sink/fan unit. The warm will make gooey the thermal interface material.

25.10.3 Fan Tachometers

Each FPGA fan has a tachometer connected to it for the detection of fan failure. If you intend to use this system in a rack or production system, you may want to monitor the fans. The fans are likely the least reliable component on the board, and may go bad. We have more.

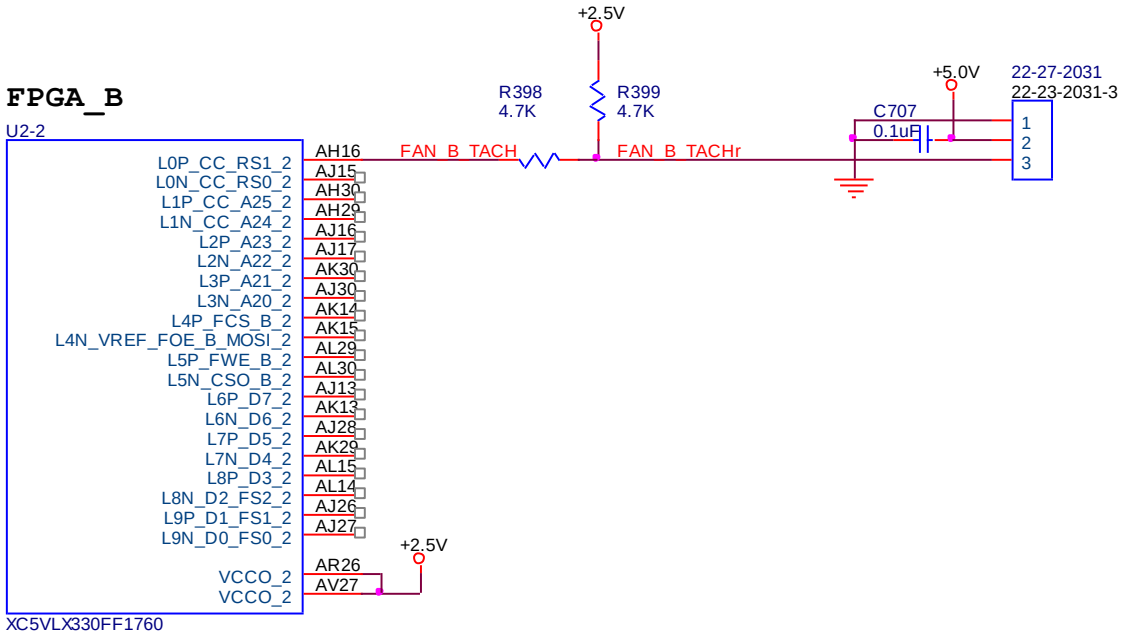


Figure 113 - Fan tachometer circuit

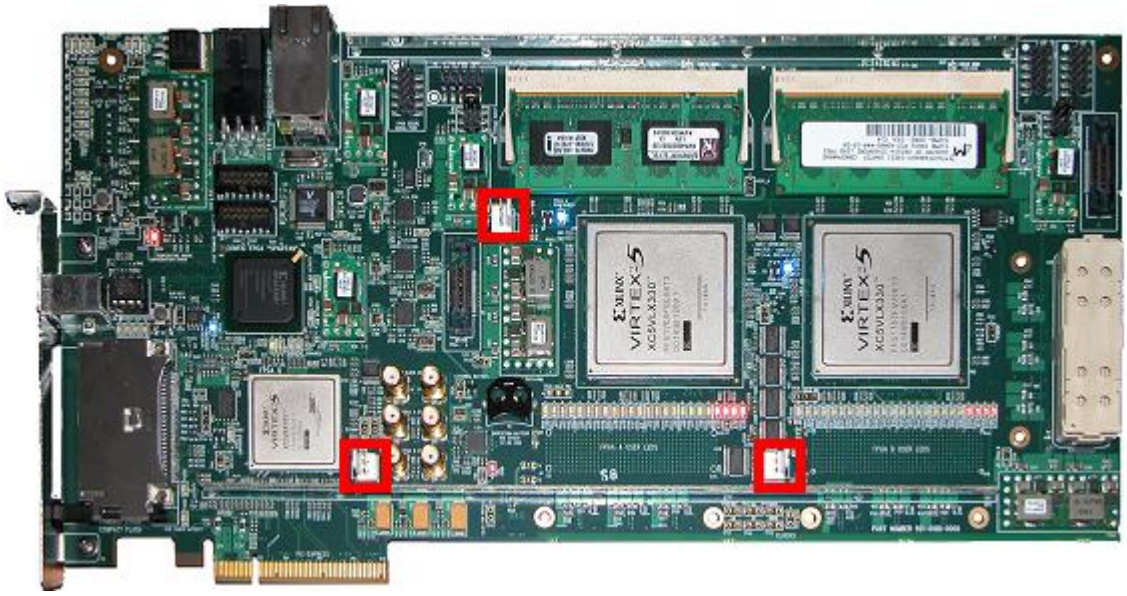


Figure 114 - Fan power locator

The fan tachometer inputs (AH16) can be LVCMOS25. The fan will produce 2 rising edges per revolution. You may need to de-bounce the signal if you intend to count the fan frequency with any precision.

Do not allow gasoline to touch the board. Do not allow dogs to chew on the board. Do not place the board under a soldering iron or on the surface of the sun.

26Connectors

This section lists all the connectors on the board

JP1	Samtec	TSM-136-01-T-DV	Change DIMM voltage
JP2	Samtec	TSM-136-01-T-DV	Change DIMM voltage
JP16	Japan		
P1	Samtec	TSM-136-01-T-DV	
P2	Samtec	TSM-136-01-T-DV	
P3	Samtec	TSM-136-01-T-DV	
P4	Samtec	TSM-136-01-T-DV	
P5	Samtec	TSM-136-01-T-DV	
P7	Samtec	TSM-136-01-T-DV	
P8	Samtec	TSM-136-01-T-DV	
J7	Molex	22-27-2031	
J15	Molex	22-27-2031	
J18	Molex	22-27-2031	
J5	Molex	87832-1420	
J6	Molex	87832-1420	
J2	JAE	MM50-200B2-1E	
J1	JAE	MM50-200B2-1E	
T1	Belfuse	0826-1X1T-23-F1	
J4	AMP/Tyco	2-5767004-2	
J8	AMP/Tyco	2-5767004-2	
J19	AMP/Tyco	2-5767004-2	
J9	Molex	67068-8000	
P5	FCI	84520102LF	
P9	FCI	84520102LF	
P10	FCI	84520102LF	
X1	AMP/Tyco	2-641260-1	
J10	Lighthorse	LTI-SASF546-P26-X1	
J11	Lighthorse	LTI-SASF546-P26-X1	
J13	Lighthorse	LTI-SASF546-P26-X1	
J14	Lighthorse	LTI-SASF546-P26-X1	
J16	Lighthorse	LTI-SASF546-P26-X1	
J17	Lighthorse	LTI-SASF546-P26-X1	
J12	Molex	53856-5070	
J3	Molex	45558-0002	
Y2	Gompf	9456-0216LC	
S1	ITT	PTS645SH50SMTRLFS	
S2	ITT	PTS645SH50SMTRLFS	
TP13	3M	923345-01-C	
TP16	3M	923345-01-C	

26.1.1 Comments

If you have a board with fewer than two FPGAs installed, connectors to which noting connects will be un-installed from the board to prevent confusion and anger.

27Mechanical

The DN9200K10PCIE8T is larger than the PCI Express specification allows, and is not guaranteed to fit into every ATX case. It will certainly fail to fit into a rack mount server enclosure. The vertical clearance with the fans installed and the ATX power connector not connector is 30mm. Lower-profile fans are available (14mm) but they may not have enough thermal performance for very power-hungry designs.

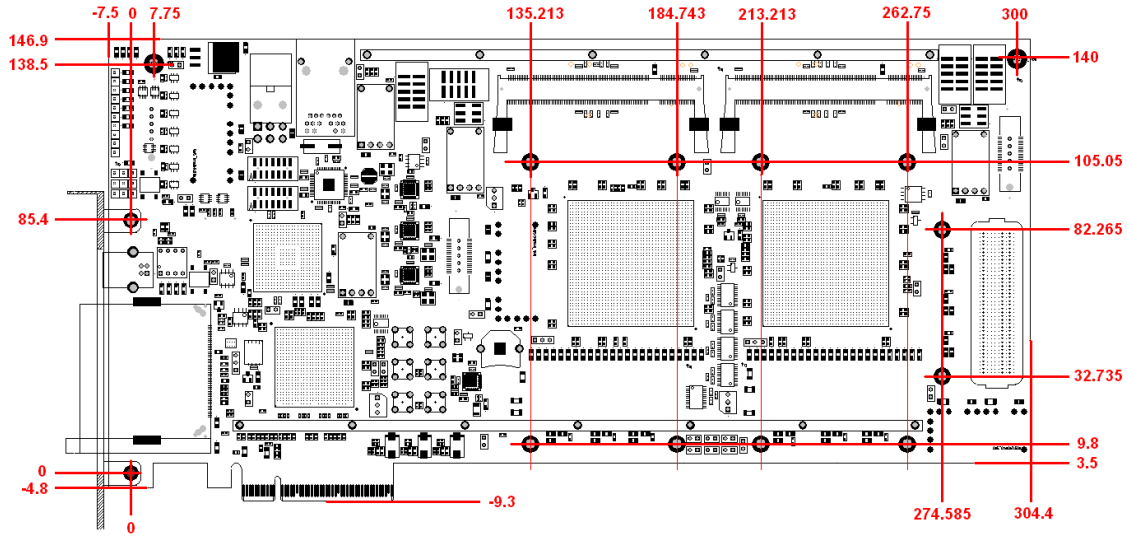


Figure 115 - Mechanical drawing

Mounting holes are all over the place. These are grounded.

Metal runners are along both edges of the board. These are for ground oscilloscope probe ground clips. You should also handle the DN9200K10PCIE8T by its ground bars to help prevent ESD damage to the FPGAs.

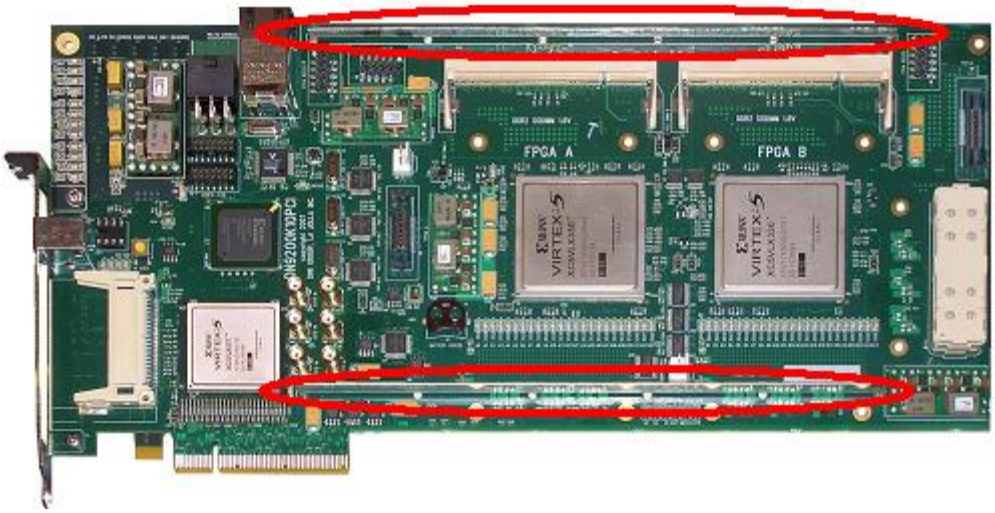


Figure 116 - Ground rail locator

28Daughtercard Headers

The daughter card expansion capability of the DN9200K10PCIE8T is provided by two FCI 'MEG-Array' family connectors. It is not compatible with the 300-pin MSA standard.

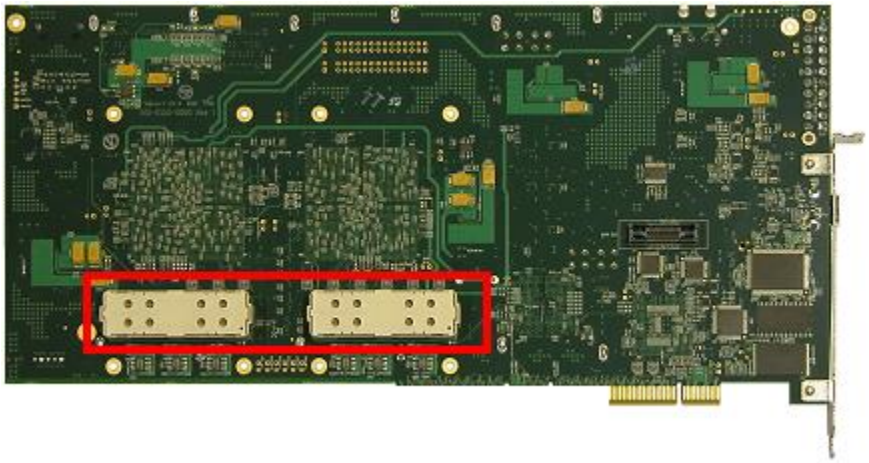


Figure 117 - Daughter card locator

Each daughtercard connector provides 186 signals (plus 4 clock signals) to its associated FPGA. The signals can be used with just about any setting of IOSTANDARD, and can be used differentially.

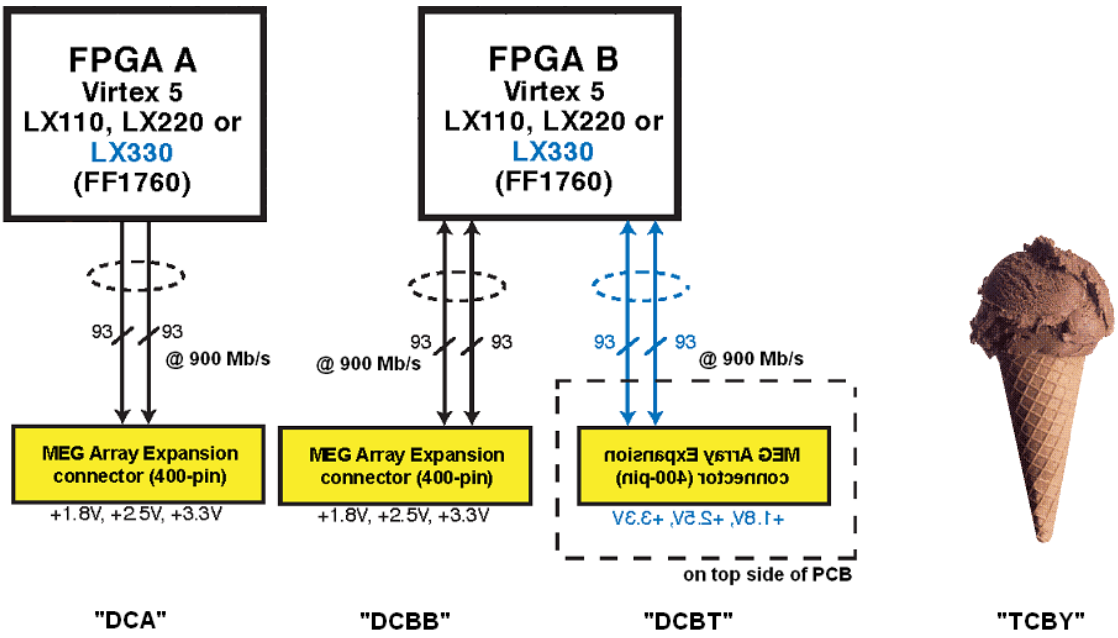


Figure 118 - Daughter card block diagram

Each daughter card header connection is arranged into three "Banks", correlating to the banks of IO on the Virtex 5 FPGA. Two "IO Banks" on the Virtex-5 FPGA connect to each one "bank" on the daughtercard connector.

This allows three different sets of voltage or timing requirements to be met on a single daughter card simultaneously. Each Bank on the daughter card is 62 signals. Each "bank" on an FPGA is 40 signals.

Other connections on the daughter card connector system include three dedicated, differential clock connections for inputting global clocks from an external source, power connections, bank VCCO power, and a buffered reset signal.

28.1 Daughter Card Physical

The connectors used in the expansion system are FCI MEG-Array 400-pin plug, 6mm, part #84520-102. This connector is capable of as much as 10 Gbs transmission rates using differential signaling.

Two daughter card expansion headers on the DN9200K10PCIE8T are located on the bottom side of the PWB. This is done to eliminate the need for resolving board-to-board clearance issues, assuming the daughter card uses no large components on the backside.

One expansion connector is provided on the front, for variety.

The "Plug" of the system is located on the DN9200K10PCIE8T, and the "receptacle" is located on the expansion board.

28.1.1 Daughter Card Locations and Mounting

The 400-pin daughtercard header is located on the bottom (solder) side near the right side of the board. Each MEG-Array header on a Dini Group product has four standard-position mountain holes. The drawing below shows the location of the daughter card header and its associated mounting holes.

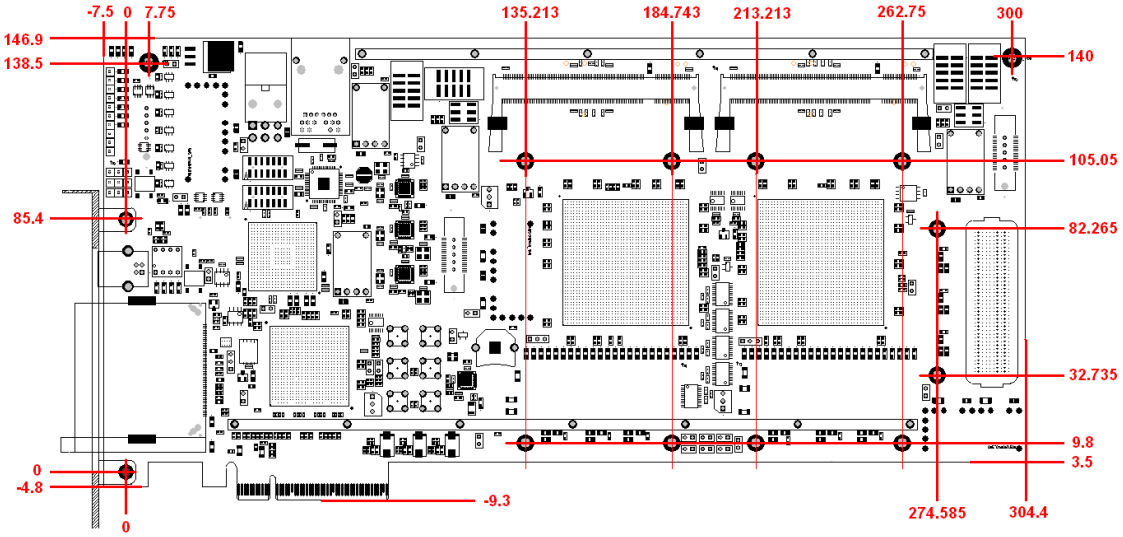


Figure 119 - Mechanical Drawing

This view of the DN9200K10PCIE8T daughter card locations is from the top of the PCB, looking through to the bottom side. The Dini Group standard daughtercard, DNMEG_OBS400 is compatible with the DN9200K10PCIE8T.

The mounting holes are designed to be used with 14mm, M3 standoffs. Dini Group has available appropriate mounting hardware on request:

Standoffs (Male-to-Female), (Part 1789)
Harwin R30-3001402
(Mouser 855-R30-3001402)
"M3 x 14mm HEX 5mmA/F Harwin Metric Spacers RoHS: Compliant. Box/100"

Big Round Nuts, (Part 1787)
LMI HN4600300
"M3 x 0.5mm

Screws, (Part 1788)
MPMS 003-0005-PH
(Digi-key H742-ND)
"SCREW MACHINE METRIC PH M3x5MM"

With this host-plate-daughter card arrangement, there is a limited Z dimension clearance for backside components on the daughter card. This dimension is determined by the daughter card designer's part selection for the MEG-Array receptacle.

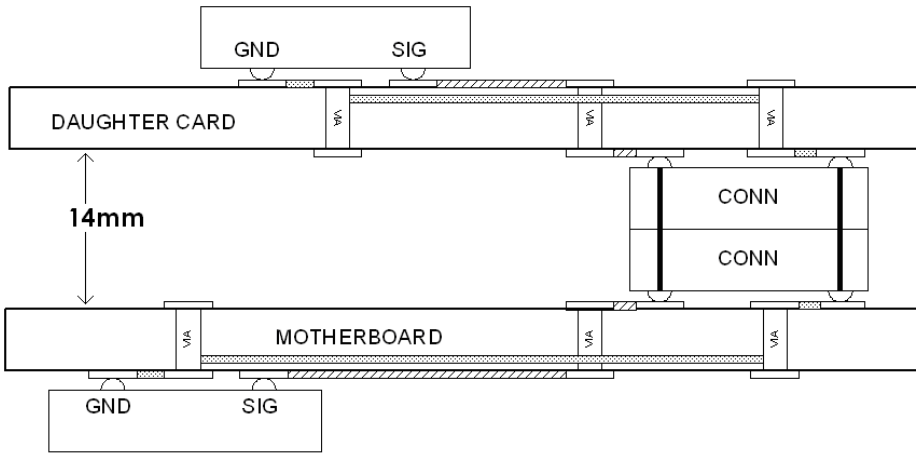


Figure 120 - Daughter card side mechanical

Note that the components on the topside of the daughter card and DN9200K10PCIE8T face in opposite directions.

28.1.1.1 DNMEG_EXT

If you need some more vertical clearance between daughtercard and DN9200K10PCIE8T (or need to install two daughtercards that interfere with each other mechanically, you can try using the DNMEG_EXT riser card.

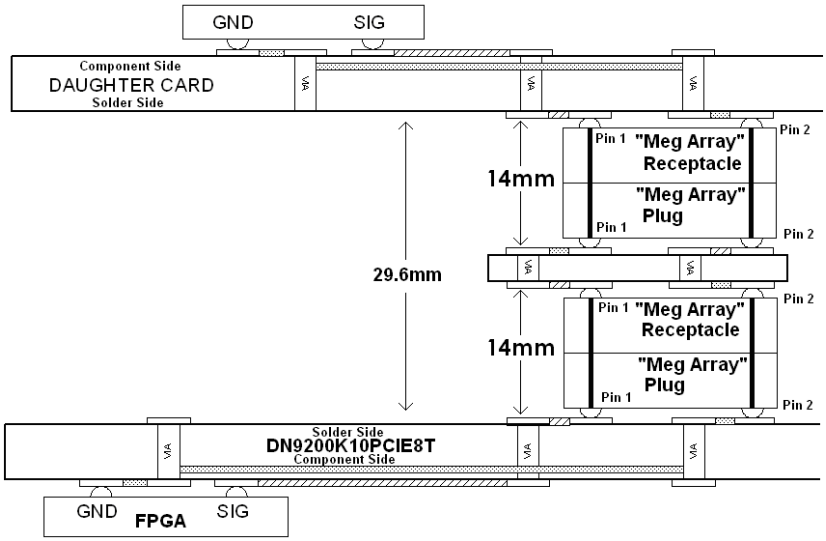


Figure 121 - DNMEG_EXT mechanical

This card extends the vertical separation between daughter card and DN9200K10PCIE8T by an additional 14mm + 0.062"

I'd also like to point out that a daughtercard designer is free to use one of three different Meg Array receptacles with different stacking heights.

28.1.2 Standard Daughtercard Size

The daughtercard mechanical provisions on the DN9200K10PCIE8T are designed to mount a hypothetical daughtercard with the dimensions given below. The "observation daughtercard", DNMEG400_OBS product conforms to these dimensions.

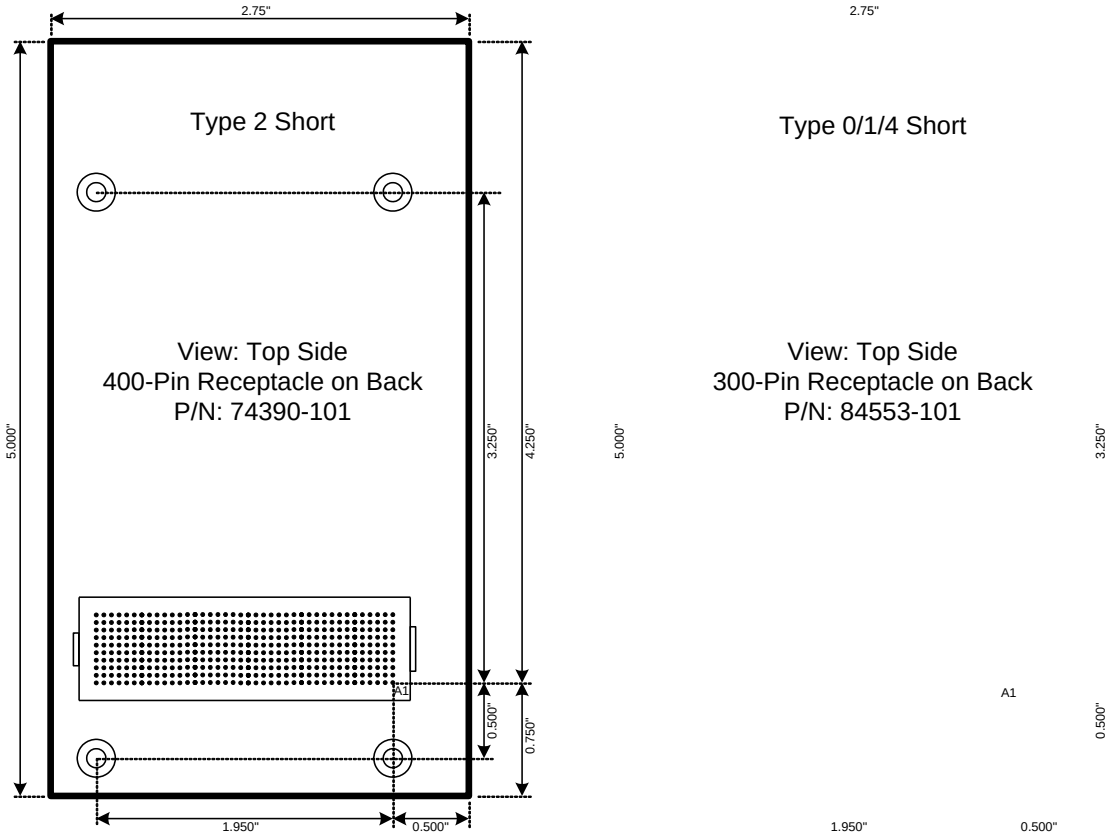


Figure 122 - Standard daughter card dimensions

The board edge constraints given above allow one daughtercard to be installed on all positions of the DN9200K10PCIE8T simultaneously. When making a daughtercard, you do not have to follow this size restriction.

28.1.3 Insertion and removal

Due to the small dimensions of the very high speed Meg Array connector system, the pins on the plug and receptacle of the Meg Array connectors are very delicate.

When plugging in a daughter card, make sure to align the daughter card first before pressing on the connector. Be absolutely certain that both the small and the large keys at the narrow ends of the Meg Array line up BEFORE applying pressure to mate the connectors!

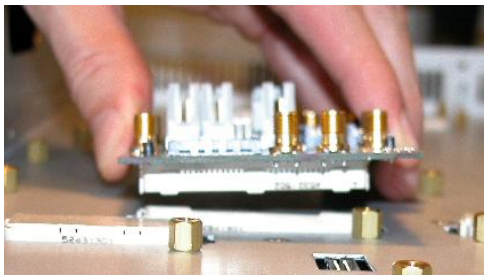


Figure 123 - Daughter card installation step 1

Place it down flat, then press down gently.

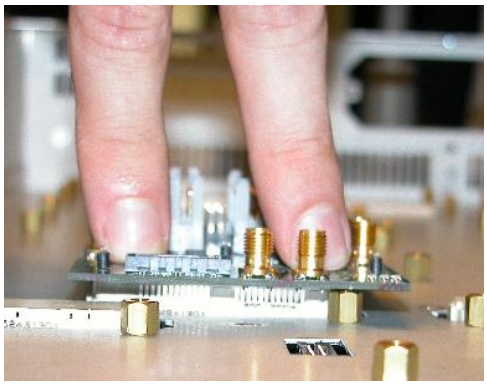


Figure 124 - Install Daughter card step 2

Mating can be started from either end. Locate and match the connector's A1 position marking [triangle] for both the Plug and Receptacle. (Markings are located on the long side of the housing.) Rough alignment is required prior to connector mating as misalignment of >0.8mm could damage connector contacts. Rough alignment of the connector is achieved through matching the Small alignment slot of the plug housing with the Small alignment key of the receptacle housing and the large alignment slot with the large alignment key. Both connector housings have generous lead-in around the perimeter and will allow the user to blind mate assemble the connectors. Align the two connectors by feel and when the receptacle keys start into the plug slots, push down on one end and then move force forward until the receptacle cover flange bottoms on the front face of the plug.

Like mating, a connector pair can be unmated by pulling them straight apart. However, it requires less effort to un-mate if the force is originated from one of the slot/key ends of the assembly. (Reverse procedure from mating) Mating or un-mating of the connector by rolling in a direction perpendicular to alignment slots/keys may cause damage to the terminal contacts and is not recommended.

28.2 Daughter Card Electrical

The daughter card pin out and routing was designed to allow use of the Virtex 5's 1.2 Gbps general purpose IO. All signals on the DN9200K10PCIE8T are all routed as differential, 50 Ω

(signal-to-ground) transmission lines. Signals can be used as single-ended also. Proper electrical levels are explained in the VCCO section.

No length-matching is done on the PCB for daughter card signals, (except between two sides of a differential pair). However, the Virtex 5 is capable of variable-delay input or output using the built-in IDELAY or ODELAY modules. A signal delay report is available here. In order to simulate a length-match, you can instantiate an IDELAY and an ODELAY element on each IO, and add a tap delay to each IO.

Signal Name	Additive Delay (ps)	Equivalent TAP value
CLK_DCA_0	525	
CLK_DCA_1	600	
DCA1P06	160	7
DCA1P10	182	7
DCA1P26	200	6
DCA1P14	200	6
DCA0P24	201	6
DCA2P27	201	6
DCA1P18_C	209	6
DCA0P20_C	210	6
DCA1P22_C	211	6
DCA2P06	216	6
DCA0P04	218	6
DCA1P25	220	6
DCA0P08	221	6
DCA2P25	224	6
DCA2P10	227	6
DCA1P17	230	6
DCA0P30	232	6
DCA1P21	234	6
DCA2P18	235	6
DCA0P03	237	6
DCA2P22	239	6
DCA2P21	240	6
DCA0P19	241	6
DCA0P18	242	6
DCA0P07	245	6
DCA1P31	247	6
DCA1P13	247	6
DCA0P11	249	6
DCA2P12	253	6
DCA0P23	255	6
DCA1P08	257	6
DCA0P22	263	6
DCA2P31	265	6
DCA1P16	270	5
DCA1P05	273	5
DCA0P16_C	275	5
DCA2P11	277	5
DCA1P29	279	5

DCA2P26	281	5
DCA0P21	281	5
DCA1P12	283	5
DCA2P13_C	289	5
DCA0P09	295	5
DCA2P05	295	5
DCA0P25	298	5
DCA2P30	300	5
DCA2P28	302	5
DCA2P03	303	5
DCA1P07	304	5
DCA0P14	309	5
DCA0P17_C	310	5
DCA2P29	310	5
DCA2P09	311	5
DCA2P14	315	5
DCA1P27	318	5
DCA2P17_C	320	5
DCA1P09	321	5
DCA1P03	333	5
DCA2P04	346	4
DCA2P08	346	4
DCA1P11	346	4
DCA2P02	353	4
DCA2P20_C	354	4
DCA1P01	355	4
DCA0P15	373	4
DCA0P12	376	4
DCA0P13_C	379	4
DCA1P23	381	4
DCA2P07	393	4
DCA1P30	393	4
DCA2P23	398	4
DCA2P01	398	4
DCA0P31	399	4
DCA0P10	400	4
DCA0P05	402	4
DCA0P29	408	4
DCA1P02	410	4
DCA1P19_C	412	4
DCA0P02	412	4
DCA2P16_C	413	4
DCA1P15_C	414	4
DCA0P01	415	4
DCA1P20	419	3
DCA1P24	422	3
DCA0P06	476	3
DCA0P26	486	3
DCA0P27	501	2
DCA2P15	509	2
DCA0P28	543	2

DCA1P04	561	2
DCA2P24	570	1
DCA2P19	638	1
DCA1P28	681	0

Signal Name	Additive Delay	Equivalent TAP value
CLK_DCBB_0	575	
CLK_DCBB_1	450	
DCBB1P02	144	9
DCBB2P04	158	9
DCBB1P18_C	167	9
DCBB1P26	167	9
DCBB2P08	174	9
DCBB1P22_C	179	9
DCBB1P21	191	8
DCBB1P01	192	8
DCBB1P29	194	8
DCBB1P25	194	8
DCBB1P06	196	8
DCBB1P14	199	8
DCBB2P12	205	8
DCBB1P05	205	8
DCBB2P03	210	8
DCBB2P16_C	213	8
DCBB1P28	214	8
DCBB0P18	223	8
DCBB0P30	231	8
DCBB2P11	233	8
DCBB2P28	233	8
DCBB2P27	237	8
DCBB0P17_C	238	8
DCBB2P14	239	8
DCBB1P03	239	8
DCBB1P30	241	8
DCBB1P31	247	8
DCBB1P10	252	8
DCBB0P22	255	8
DCBB0P21	256	8
DCBB1P09	257	7
DCBB1P13	257	7
DCBB1P12	258	7
DCBB1P04	270	7
DCBB1P07	273	7
DCBB2P30	275	7
DCBB2P15	278	7
DCBB2P24	280	7
DCBB2P29	283	7
DCBB1P11	284	7
DCBB2P22	287	7
DCBB2P31	288	7

DCBB1P27	290	7
DCBB2P23	293	7
DCBB2P20_C	298	7
DCBB1P16	298	7
DCBB0P13_C	300	7
DCBB0P09	306	7
DCBB0P10	307	7
DCBB0P25	307	7
DCBB2P26	315	7
DCBB0P06	317	7
DCBB1P20	320	7
DCBB0P23	320	7
DCBB0P24	323	7
DCBB2P01	329	7
DCBB2P18	337	6
DCBB1P08	337	6
DCBB2P17_C	339	6
DCBB1P17	340	6
DCBB2P25	342	6
DCBB0P15	345	6
DCBB2P02	351	6
DCBB0P20_C	351	6
DCBB0P02	353	6
DCBB0P16_C	364	6
DCBB0P28	367	6
DCBB0P27	371	6
DCBB0P05	375	6
DCBB2P19	376	6
DCBB2P07	376	6
DCBB2P21	385	6
DCBB2P13_C	385	6
DCBB0P01	390	6
DCBB1P24	390	6
DCBB0P26	390	6
DCBB1P23	397	6
DCBB0P29	405	6
DCBB0P11	410	5
DCBB1P15_C	434	5
DCBB1P19_C	436	5
DCBB0P03	441	5
DCBB0P14	474	5
DCBB2P09	504	4
DCBB2P05	513	4
DCBB0P19	539	4
DCBB2P10	554	4
DCBB0P07	574	3
DCBB0P12	574	3
DCBB0P08	585	3
DCBB2P06	594	3
DCBB0P04	783	0
DCBB0P31	863	0

Signal Name	Additive Delay	Equivalent TAP value
CLK_DCBT_0	730	
CLK_DCBT_1	658	
DCBT2P30	318	10
DCBT1P11	319	10
DCBT1P15_C	322	10
DCBT2P23	330	10
DCBT1P28	349	9
DCBT1P17	350	9
DCBT1P25	354	9
DCBT1P19_C	359	9
DCBT1P29	359	9
DCBT2P26	361	9
DCBT2P21	362	9
DCBT2P09	362	9
DCBT2P13_C	365	9
DCBT2P24	369	9
DCBT2P15	371	9
DCBT2P17_C	377	9
DCBT2P28	380	9
DCBT1P13	383	9
DCBT0P01	384	9
DCBT2P16_C	385	9
DCBT1P21	387	9
DCBT2P18	387	9
DCBT2P22	391	9
DCBT0P10	392	9
DCBT2P19	395	9
DCBT2P02	395	9
DCBT1P05	396	9
DCBT2P04	396	9
DCBT0P17_C	398	9
DCBT2P08	399	9
DCBT2P11	402	9
DCBT2P12	403	9
DCBT1P18_C	405	9
DCBT0P21	409	9
DCBT0P13_C	412	9
DCBT2P10	414	9
DCBT1P08	418	8
DCBT1P07	418	8
DCBT0P06	427	8
DCBT0P25	432	8
DCBT2P07	435	8
DCBT0P18	438	8
DCBT0P26	439	8
DCBT0P14	439	8
DCBT1P09	439	8
DCBT1P10	440	8

DCBT2P03	441	8
DCBT2P27	455	8
DCBT0P30	460	8
DCBT0P02	462	8
DCBT0P15	464	8
DCBT0P29	464	8
DCBT2P20_C	465	8
DCBT2P05	466	8
DCBT2P25	472	8
DCBT1P12	480	8
DCBT0P03	483	8
DCBT1P03	491	7
DCBT0P12	494	7
DCBT0P11	497	7
DCBT1P04	499	7
DCBT0P22	520	7
DCBT1P02	535	7
DCBT1P01	535	7
DCBT1P14	537	7
DCBT2P14	540	7
DCBT1P06	552	7
DCBT0P07	553	7
DCBT0P16_C	560	7
DCBT0P28	570	6
DCBT0P19	571	6
DCBT0P20_C	577	6
DCBT1P22_C	581	6
DCBT2P29	596	6
DCBT1P26	598	6
DCBT0P23	601	6
DCBT0P27	610	6
DCBT1P27	625	6
DCBT0P24	663	5
DCBT0P04	679	5
DCBT0P08	686	5
DCBT1P31	693	5
DCBT1P20	724	4
DCBT2P06	759	4
DCBT2P01	771	4
DCBT0P09	857	3
DCBT2P31	859	3
DCBT0P31	864	3
DCBT1P16	865	3
DCBT1P30	872	2
DCBT1P24	885	2
DCBT0P05	953	1
DCBT1P23	1053	0

28.2.1 Pin assignments

The pin out of the DN9200K10PCIE8T expansion system was designed to reduce cross talk to manageable levels while operating at full speed of the Virtex 5. The ground to signal ratio of the connector is 1:1. General purpose IO is arranged in a GSGS pattern to allow high speed single-ended or differential use. On the DN9200K10PCIE8T (host), these signals are routed as loosely-coupled differential signals, meaning when used differentially, they benefit from the noise-resistant properties of a differential pair, but when used single-ended-ly, do not interfere with each other excessively.

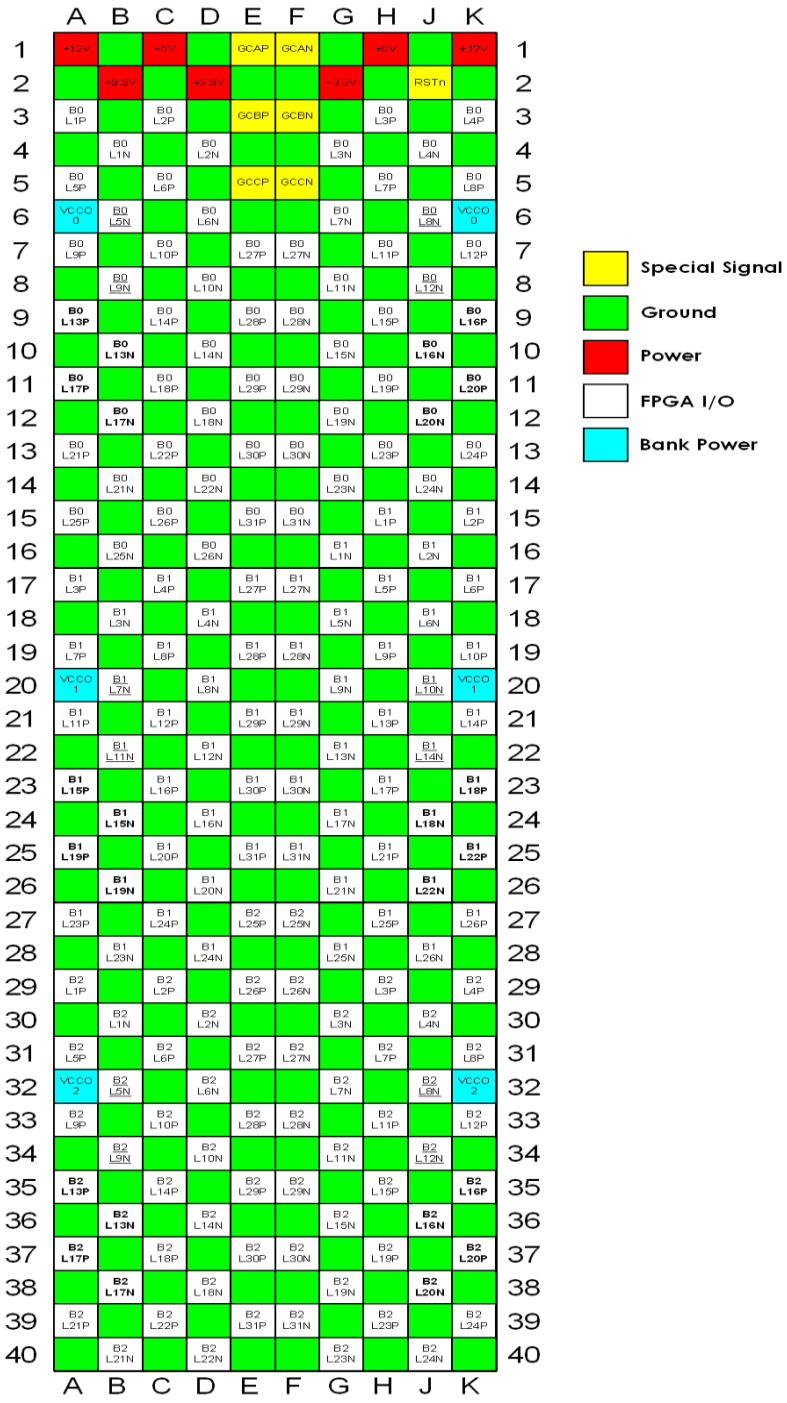

A B C D E F G H J K
1 +12V +5V GCAP GCAN +5V +12V
2 +3.3V +3.3V +3.3V RSTn
3 B0 L1P B0 L2P GCBP GCBN B0 L3P B0 L4P
4 B0 L1N B0 L2N B0 L3N B0 L4N
5 B0 L5P B0 L6P GCCP GCCN B0 L7P B0 L8P
6 VCCO 0 B0 L5N B0 L6N B0 L7N B0 L8N VCCO 0
7 B0 L9P B0 L10P B0 L27P B0 L27N B0 L11P B0 L12P
8 B0 L9N B0 L10N B0 L11N B0 L12N
9 B0 L13P B0 L14P B0 L28P B0 L28N B0 L15P B0 L16P
10 B0 L13N B0 L14N B0 L15N B0 L16N
11 B0 L17P B0 L18P B0 L29P B0 L29N B0 L19P B0 L20P
12 B0 L17N B0 L18N B0 L19N B0 L20N
13 B0 L21P B0 L22P B0 L30P B0 L30N B0 L23P B0 L24P
14 B0 L21N B0 L22N B0 L23N B0 L24N
15 B0 L25P B0 L26P B0 L31P B0 L31N B1 L1P B1 L2P
16 B0 L25N B0 L26N B1 L1N B1 L2N
17 B1 L3P B1 L4P B1 L27P B1 L27N B1 L5P B1 L6P
18 B1 L3N B1 L4N B1 L5N B1 L6N
19 B1 L7P B1 L8P B1 L28P B1 L28N B1 L9P B1 L10P
20 VCCO 1 B1 L7N B1 L8N B1 L9N B1 L10N VCCO 1
21 B1 L11P B1 L12P B1 L29P B1 L29N B1 L13P B1 L14P
22 B1 L11N B1 L12N B1 L13N B1 L14N
23 B1 L15P B1 L16P B1 L30P B1 L30N B1 L17P B1 L18P
24 B1 L15N B1 L16N B1 L17N B1 L18N
25 B1 L19P B1 L20P B1 L31P B1 L31N B1 L21P B1 L22P
26 B1 L19N B1 L20N B1 L21N B1 L22N
27 B1 L23P B1 L24P B2 L25P B2 L25N B1 L25P B1 L26P
28 B1 L23N B1 L24N B1 L25N B1 L26N
29 B2 L1P B2 L2P B2 L26P B2 L26N B2 L3P B2 L4P
30 B2 L1N B2 L2N B2 L3N B2 L4N
31 B2 L5P B2 L6P B2 L27P B2 L27N B2 L7P B2 L8P
32 VCCO 2 B2 L5N B2 L6N B2 L7N B2 L8N VCCO 2
33 B2 L9P B2 L10P B2 L28P B2 L28N B2 L11P B2 L12P
34 B2 L9N B2 L10N B2 L11N B2 L12N
35 B2 L13P B2 L14P B2 L29P B2 L29N B2 L15P B2 L16P
36 B2 L13N B2 L14N B2 L15N B2 L16N
37 B2 L17P B2 L18P B2 L30P B2 L30N B2 L19P B2 L20P
38 B2 L17N B2 L18N B2 L19N B2 L20N
39 B2 L21P B2 L22P B2 L31P B2 L31N B2 L23P B2 L24P
40 B2 L21N B2 L22N B2 L23N B2 L24N
A B C D E F G H J K
Special Signal
Ground
Power
FPGA I/O
Bank Power

Figure 125 - Daughter card pinout diagram

All high-speed signals on the DN9200K10PCIE8T, including daughter card signals, are routed against a ground potential reference plane. When creating a daughter card, it is recommended that these signals remain against a ground plane to maintain trace impedance.

The central columns of the connector pin out use a closely coupled, differential pair pin arrangement, which is uniformly surrounded by ground pins.

Below is a graphic representation of the pin assignments for the 400-pin connectors. Note that this is a view from the backside of the connector. The green boxes represent ground connections.

Special purpose pins are described below.

28.2.2 CC, VREF, DCI

Some of the signals connected to the daughter card expansion headers are "clock-capable"; the inputs on the Virtex 5 FPGA can be used for source-synchronous clocking. In the schematic and customer netlist on the user CD, these pins contain a "_C" in the pin name.

Pins declared in the above diagram that are underlined are connected to "VREF" pins on the Virtex 5 FPGA. These FPGA pins are used to supply a voltage reference used as the threshold voltage for the signals on that bank. The use of these pins is only necessary when using threshold standards, such as SSTL.

DCI is used on all FPGA IO banks connected to a daughter card header. The reference resistance is 50Ω. Each Virtex 5 bank that is connected to a header DCI in enabled.

28.2.3 Global clocks

The daughter card pin out defines 6 clock output pins. These clock outputs are intended to be used a 3 differential signals (LVDS). Two clock signals GCA and GCB connect to the "GC" clock inputs on the FPGA. These clocks can be used only by the FPGA that is associated with the header.

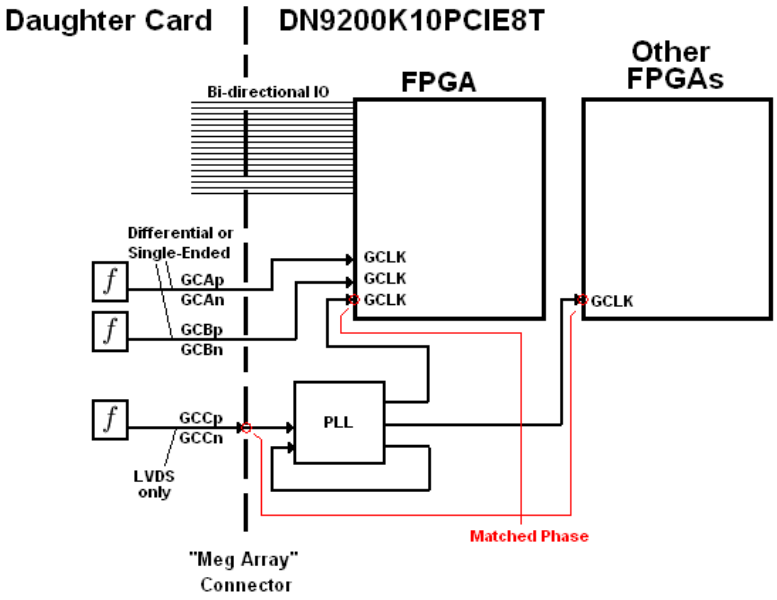


Figure 126 - Daughter card clock pin functions

The GCC(p/n) signal driven from each FPGA connects to a global clock buffer and can be used by all of the FPGAs on the DN9200K10PCIE8T. (EXT0 and EXT1 networks). Since the two daughter cards B share the same clock network (EXT1), only one of these two daughtercards can drive a global clock at one time. In order to have a phase match between the GCC clock pin at the clock input pins on the FPGA, the PLL on the EXT clock network must be enabled and set to the proper frequency. Also note that the PLL cannot account for delay on the daughtercard between the frequency source and the GCC pin.

28.2.4 Timing and Clocking

Signal from the FPGAs to the daughtercard connector are not length-matched. There is a length-report above somewhere.

Each daughtercard has a global clock output pair "DCCLKCp/n". This LVDS output is distributed on the DN9200K10PCIE8T to all Virtex-5 FPGAs. The clock buffer on the host board is designed to deliver the clock edge to all FPGA synchronized with the CCLK pin on the daughtercard header. The daughtercard is expected to distribute clocks on it so that ICs on the daughtercard receive the clock signal synchronized with the pin on the daughtercard header. In this way, the host and daughter boards should be able to communicate synchronously with equal, large IO periods in each direction.

There are at least four methods of communicating FPGA-to-FPGA across the daughtercard interface.

28.2.4.1 Local Synchronous

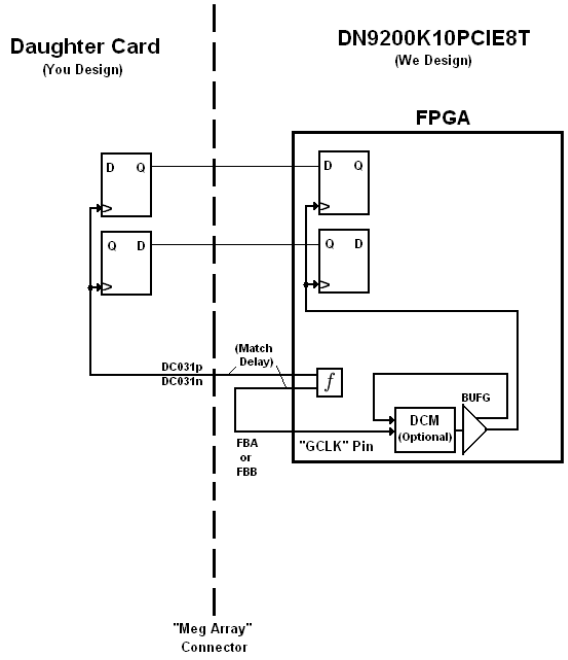


Figure 127 - Daughtercard clocking local

The daughtercard generates a clock and drives it over the GCAp/n or GCBp/n clock pins to the host board FPGA. The daughtercard drives a synchronized clock to the logic on the daughtercard, adding 0.5ns delay to account for the trace delay on the DN9200K10PCIE8T. The host FPGA will use a DCM in zero-delay mode, and the logic on the daughtercard should have a low clock-to-out and setup times. (or use a DCM). This method has the disadvantage of only allowing the one FPGA attached to the daughtercard to use this frequency. To communicate globally across the DN9200K10PCIE8T, the user would have to pass the data across clock domains, or add another layer of DCMs to adjust the daughtercard skew to match the rest of the board.

28.2.4.2 Global Synchronous

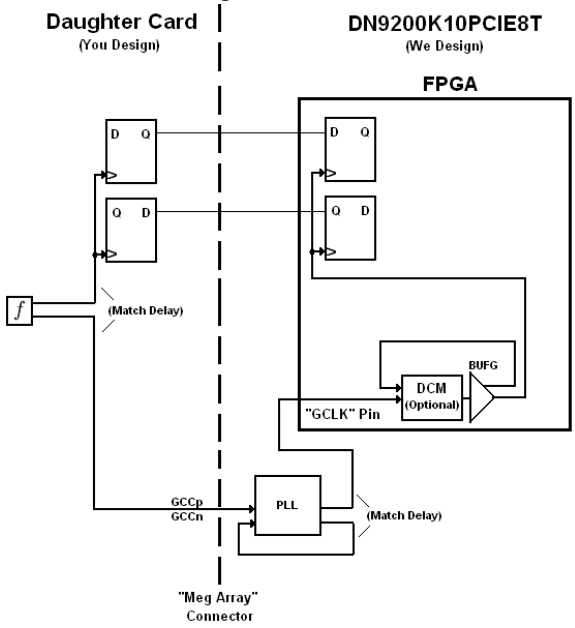


Figure 128 - Daughter card clocking global

The daughter card generates a clock and drives it over the GCCp/n pins to the DN9200K10PCIE8T host board. The user will select the daughtercard source for either the EXT0 or EXT1 networks as appropriate. The user sets the EXT0 or EXT1 network into zero-delay mode. See EXT0 and EXT1 in the clocking section. The disadvantage of this method is that the EXT0 or EXT1 network must be used, and that the zero-delay configuration has to be calculated by looking at the datasheet, or by using the CompactFlash card. DCARD instruction. The advantage is that the entire system can be operated on a single clock domain.

Zero-delay on the DN9200K10PCIE8T is allowed by enabling PLL devices (zero-delay buffers) connected to the GCC pins of each daughtercard header. To allow for a very wide range of clock frequencies sourced from the daughtercard, the PLL bandwidth of these buffers must be manually set. This can be done via USB, PCIe or Compact Flash. The PLL can also be bypassed, allowing a global system-synchronous clock to be used without configuring this PLL. When using this method, the daughtercard will have no information about the phase of the clock arrival at the FPGAs, and the FPGA will have to drive a clock back to the daughtercard.

28.2.4.3 Source Synchronous

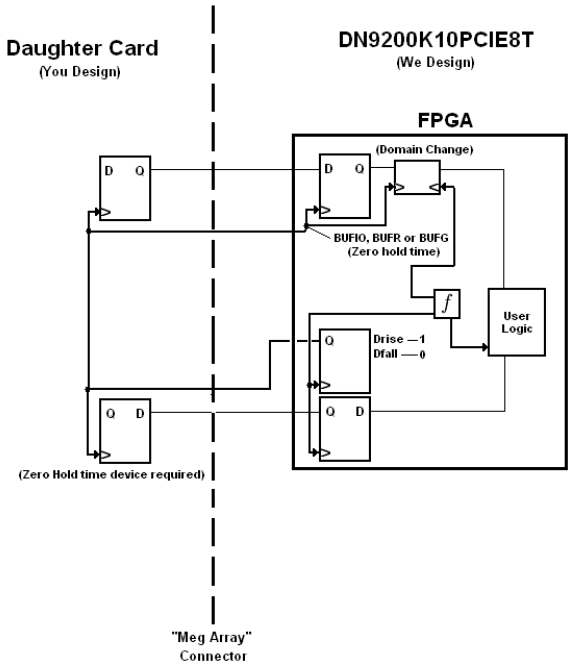


Figure 129 - Daughter card clocking source synchronous

The daughtercard drives a clock into the CC pins of the daughtercard connector. This clock is used to latch IOs. This method should be used for frequencies exceeding 150 MHz, because the phase-tolerance of the Virtex 5 FPGA and the clock buffer devices on the DN9200K10PCIE8T EXT0 and EXT1 signals will prevent a reliable system-synchronous design at high speeds. This method has the advantage of being the fastest design technique. Additionally, no DCMs or PLL are required. This is the only method that works with a non free-running clock.

28.2.4.4 Skewed Clocks

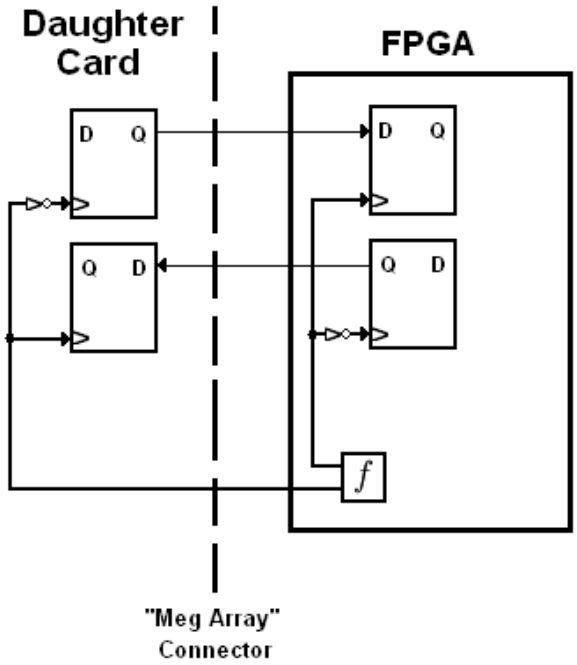


Figure 130 - Daughter card clocking skew tolerant

It is possible to create a synchronous IO system that is tolerant of phase differences between link partners. In the above example, outputs are clocked on the falling edge of the clock, and inputs are clocked on the rising edge of the clock.

The advantage of this system is that it is the simplest clock network; it does not require a free-running clock (no DCM or PLL). The disadvantage is that is requires the use of DDR flip-flops, which may not be available on all parts (then you would need to drive two clocks to the daughter card out of phase from each other). You would also have to learn how to specify timing parameters within the FPGA from rising-edge to falling edge of a clock.

Unless you are willing to use a non-50% duty cycle clock, this method's maximum frequency is exactly half that of the fully-synchronous methods.

28.2.5 Incorrect Clocking Methods

Sometimes people incorrectly create a daughtercard clock network. Usually, they don't notice their mistake, because the errors will only show up right before the project deadline.

28.2.5.1 Clock Forwarding

You may be thinking, "It's 4 PM and I want to go home." But outputting a clock from the FPGA and using it to clock in data on the daughtercard will in most cases result in a hold-time violation.

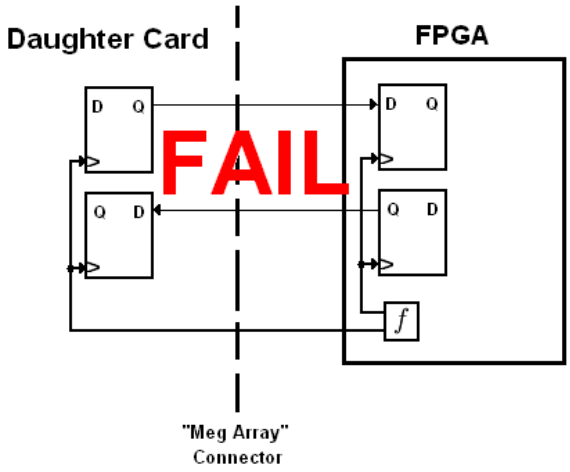


Figure 131 - Daughter card Clock forwarding fail

If you do this, you have to slow down your clock somehow. You can use external feedback, ODELAY elements or glue. Violating hold is one of the most humiliating experiences that a young engineer will ever face.

28.2.5.2 Cascading PLLs

If you try to use the "global synchronous" clock method, and then use a DCM to try to match the phase to some external clock, you will have something like is shown below.

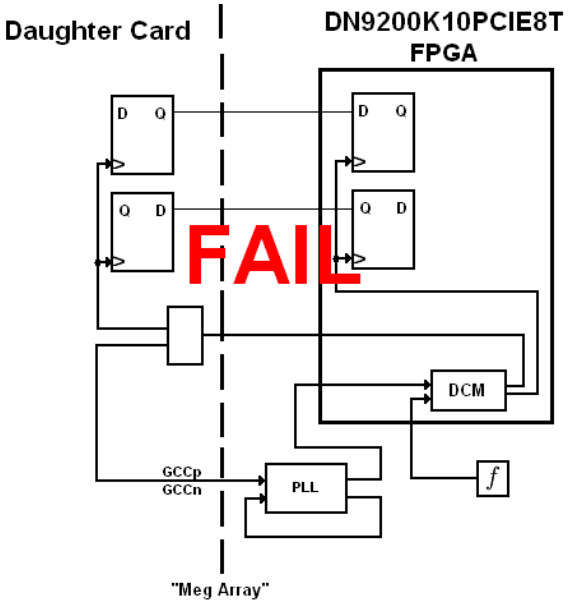


Figure 132 - Daughter card clocking PLL cascade fail

In this diagram, one PLL is within the feedback loop of another PLL. This may or may not result in harmonic instability.

28.2.6 Power and Reset

The +3.3V, +5.0V and +12V power rails are supplied to the Daughter card headers. Each pin on the MEG-Array connector is rated to tolerate 1A of current without thermal overload. Most of the power available to daughter cards through the connector comes from the two 12V pins, for a total of 24W. Each power rail supplied to the Daughter card is fused with a reset-able switch. Daughter cards are required to provide their own power supply bypassing and onrush current limiting.

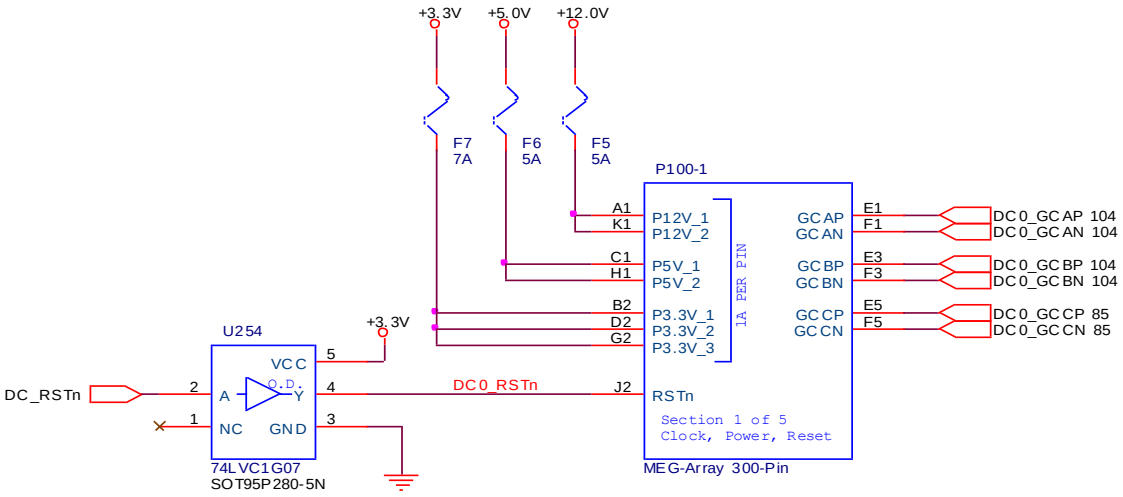


Figure 133 - MEG Array power circuit

The RSTn signal to the daughter card is an open-drain, buffered copy of the SYS_RST# signal. It is also asserted when the User Reset is active. When RSTn is de-asserted, the +3.3V, +5.0V and +12V power rails are guaranteed to be within the DN9200K10PCIE8T tolerance. If there are additional power requirements, the daughter card is required to ensure these.

28.2.7 VCCO Voltage

The daughter card is required to provide a voltage on the VCCO pin on the connector. This voltage is used on the DN9200K10PCIE8T to power the FPGA IOs that are connected with that daughter card. In this way, the daughter card can control what voltage the interface will use.

Each bank of the connector (B0, B1, or B2) uses a separate VCCO pin, and can have a different voltage applied to it. When designing a daughter card, you must determine the current requirements for the DN9200K10PCIE8T and supply enough current capacity on these pins.

The VCCO voltage impressed by the daughter card should be less than 3.75V to prevent damage to the Virtex 5 IOs connected to that daughter card. Additionally, the voltage applied to the header pins from a daughtercard or external source, should be equal to or less than the VCCO voltage of the bank that contains the IO. For example, a 2.5V daughtercard (one that uses 2.5V on each VCCO pin) should not drive a 3.3V signal onto the daughtercard pins.

28.2.8 VCCO bias generation

Since a daughter card will not always be present on a daughter card connector, a VCCO bias generator is used on the motherboard for each daughter card bank to keep the VCCO pin on the FPGA within its recommended operating range. The VCCO bias generators supply +1.2V to the VCCO pins on the FPGAs, and are back-biased by the daughter card when it drives the VCCO rails.

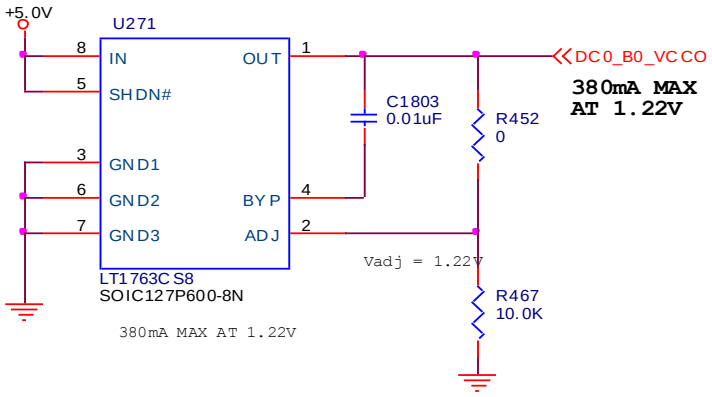


Figure 134 - MEG Array bias circuit

The output voltage of this regulator can be adjusted if needed. This will require changing the resistors on the ADJ pin of the regulators. The bias regulators can provide up to 1.5A of current. Some low-speed designs may not need more than this. Dini Group recommends placing the IO voltage regulators on the daughtercards, because this does not require modification of the DN9200K10PCIE8T.

28.3 Rolling your own daughtercard

Small quantities of the connectors required for building a daughtercard can be obtained at cost or free from the Dini Group.

The design files (PADS power PCB, schematic and Gerbers) for some example daughter cards are on the website.

If you need help designing a daughtercard, we will be happy to review your schematic for errors. Send it. Here is a totally incomplete list of stuff that we found wrong with people's daughtercards that they sent in:

- They used the schematic symbol and part footprint from the base-board when designing a daughtercard, so that pin A1 connected to pin A40 and pin K1 connected with K40.

- They provide a clock to GCC that is single-ended.

- They do not provide a voltage to +VIO0, +VIO1 and +VIO2.

- They send a clock to the FPGA into a standard IO and not a GCLK pin.

- They connected a power rail (+5V, +12V or +3.3V) to both the daughtercard and to an external power connector or a regulator on the daughter card. Dini board does not like this at all.

- They used graham crackers and peanut butter instead of FR4 and copper to save money.

- They drive a clock either from the daughtercard to the base-board or from the base board to the daughter card without accounting for clock skew. Hold time violations abound.

29Troubleshooting

29.1 The board is dead

If the board doesn't respond over USB or PCI Express it may be stuck in reset. When this happens, a red LED labeled "SYS RESET" or "HARD RESET" (near the USB connector) is on. This is usually the result of a power failure. You can see which of the voltages is causing the problem by looking at the line of red LEDs along the left edge of the board. One will be lit for each power that has failed.

- Measure 12V with a multi-meter. It should be above 11.3V

- 12V may be unstable. Connect an old hard drive to one of the 4-pin connectors on the power supply.

- The board requires the 6-pin PCI Express graphics power connector, even when installed in a PCI Express slot.

29.2 The board does not respond over PCI Express

Check first that the board is not in reset, as described above. Next, see if the blue LED next to FPGA Q is on. This LED shows whether FPGA Q is configured. If it is not configured, then there could be a problem with the Flash programming file. You can see if this FPGA will program using USB or a JTAG cable.

If the FPGA is programmed with a bitfile other than the provided "PCI Express full function endpoint now with DMA™", then you are on your own.

Otherwise, check the Windows device manager. If and "unknown device" appears on PCI Express, then there is a problem with the driver.

If the board appears to work, except all PCI transactions always respond with 0xFFFFFFFF, then the board lost its marbles. Check the lowest offsets of BAR0. If these respond with 0xFFFFFFFF then the board ate it hard. If this range works, but BAR2 doesn't work, then maybe you've just uncovered a bug in the FPGA A code.

29.3 The board does not respond over USB

If the provided software doesn't seem to be able to communicate with the board, first check that the board is not in reset (above). If it is not in reset, see if, in Windows, the board appears in device manager. If the device appears as an "Unknown Device" then the driver may not have been installed, or installed improperly. From device manager, you can see what the Vendor and Device ID of the device are. If they are both 0000, 0000 there may be a hardware problem.

Also see if the board is appearing as a some kind of Audio Device, then there is a device conflict. Call us. There is some way to fix this.

If the board is not in reset, but it still does not appear over USB, check the RS232 serial "MCU" output when the board powers on. If it stops before getting to the "main menu" then it has detected a problem and stopped before enabling USB. Send us the terminal capture.

29.4 The FPGAs won't program

First, connect the RS232 terminal and restart the board. Usually, when an FPGA fails to program, the configuration section will detect the problem and print an error message to this terminal. Common problems the configuration section might report are:

- The syntax in the main.txt file is incorrect
- The bit file on the CompactFlash card is for the wrong type of FPGA.

If the DN9200K10PCIE8T reports about one or more FPGAs that "DONE did not go high", then there is a problem with the bit file. The bit file may have been generated using bitgen options that are not compatible with the DN9200K10PCIE8T.

See if the FPGAs will configure using USB, PCIe or JTAG.

When you contact Dini Group for support, we will need a capture of the RS232 terminal output.

29.5 My design doesn't do anything

Make sure that the clock your design uses is running. Output the clock to an LED and probe it with an oscilloscope.

Check the pinout in your constraint file. Check the .PAR report file to make sure that 100% of your IOBs used have LOC constraints. There is never a reason not to constrain an IO.

Use the .PAD report to make sure your constraints were all applied. Some situations may cause constraints to be ignored.

Double-check that the connections match between your FPGA pins and the daughtercard pins using the schematic.

If "MainBus" interface is not working, make sure that none of the other FPGAs are driving those MB pins.

Make sure that the "Unused IOBs" option in bitgen is set to "Float

Check for Timing errors in the timing report

Route the clock signal to a pin and observe it with an oscilloscope.

29.6 The DCMs won't lock

1) The DCMs are required to be set in a frequency mode compatible with the frequency of the reference clock input. Check the following attributes of the DCMs.
DFS_FREQUENCY_MODE
DFS_PERFORMANCE_MODE

2) All clock inputs of the DCM are required to be stable for a certain number of microseconds before releasing the DCMs reset signal. If you are generating the reference clock from an FPGA (or another DCM), you will need to build a delayed-reset circuit to reset the second DCM.

3) Make sure the global clock you are using is being received with an LVDS receiver, not a single-ended one. Make sure the DIFF_TERM attribute is turned on (especially low frequency clocks).

29.7 It's so weird… It's like sometimes when I program my FPGAs, the signals between the FPGAs are delayed by one clock cycle. Then, when I hit the reset button, sometimes it starts working again.

Are you sending a high-speed clock to two FPGAs, them dividing the frequency in each FPGA? This doesn't work. Think about it for a second.

29.8 My pacemaker stops working when I increase the clock frequency

Make sure you have already paid the invoice.

29.9 The signal on my board is going bat crazy on my oscilloscope

Make sure the ground clip is attached to the probe.

If there is an oscillation on the signal at 60Hz, there is a problem with the oscilloscope setup.

Capture the oscilloscope view and email it to support@dinigroup.com.

If you zoom too far out on a signal, it will look like a normal signal, except that the trigger won't work and the signal will look crazy and periodic. Just zoom in like 1000 times.

If you have two oscilloscope probes and they their cables are running next to each other to the oscilloscope, you will see one signals bleeding onto the other signal. You can see if this is happening because the signals will become stronger when you grab both cables and let them couple through your hand.

Chapter 5: Reference Design

This chapter introduces the DN9200K10PCIE8T Reference Design, including information on what the reference design does, how to build it from the source files, and how to modify it for another application. This sentence has never been read.

1 Purpose

The purpose of the reference design is to demonstrate how one might implement most of the hardware capabilities of the board, to provide an example project for testing the design flow, and to test for electrical connectivity errors on the board.

While the reference design or parts of it might be useful as a starting point for your project, it is not really a product, so helping you modify the reference design to suit your needs is not within the scope of support for your board. See diagram below.

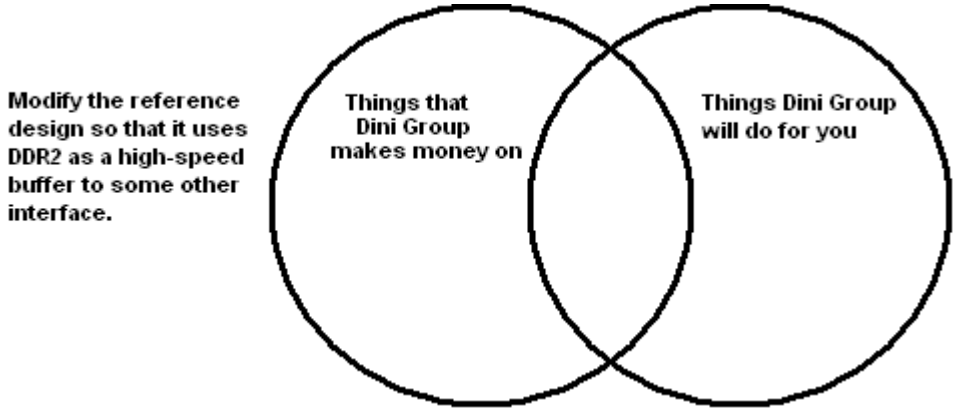

Modify the reference design so that it uses DDR2 as a high-speed buffer to some other interface.
Things that Dini Group makes money on
Things Dini Group will do for you

Figure 135 - Dini Group corporate strategy diagram

1.1 Interfaces used by reference design

The interfaces that the Dini Group design uses the following interfaces:

DDR2 Memory
PCI Express w/DMA support
USB
Main Bus
LEDs
User ("Reset") Button
Global Clock networks

1.2 Interfaces not used by the reference design

The following interfaces are not used by any reference design that Dini Group provides to users. These interfaces are fully tested, and we might even be able to give you bit files and test procedures for them.

Ethernet
Daughter Cards
External Clock inputs
RS232 (Serial Port)

2 Hardware Tests

The provided bit files and software is suitable for testing most of the hardware interfaces on your board. Some hardware tests require test fixtures, and these are not provided.

2.1.1 Testing PCI Express interface

Install the board into a windows machine in a PCI Express x16 or x8 slot (other slots will cause the test to erroneously report a failure). Turn on the machine. Run the provided executable aetest_wdm.exe. From the main menu, select "production tests" and then "PCI test".

The test should report PASS or FAIL.

2.1.2 Testing FPGA-to-FPGA interconnect

To test the FPGA interconnect, you will need to run the "one-shot test". This is a feature of the windows program USB Controller.exe. Turn on the board and connect it to a windows computer over USB.

From the "settings/info" menu, select "one shot test". Enter in one of the text boxes the path to your user CD where the bit files are kept. Unselect "DDR" from the test options, so that only interconnect is tested.

2.1.3 Testing DDR2 Interfaces

Turn on the board and connect it to a windows machine.

To test the DDR2 interface(s), configure an FPGA which has a DDR2 interface with the "Main" reference design. Install a DDR2 SODIMM into the socket of the FPGA.

In USB Controller, click the "enable USB communication" button. Then, set the global clock networks to the following frequencies:

G0 450 MHz
G1 250 MHz
G2 200 MHz

The frequency of network G1 determines the DDR2 frequency of operation. From the "settings/info" menu, select "Test DDR". In the dialog box, select the FPGA which is configured. The test will report PASS or FAIL.

2.1.4 Testing USB

USB can be tested by running the DDR2 test, or by configuring FPGAs over USB.

2.1.5 Testing Ethernet

This test can be performed by the user, however bit files are not provided. If you suspect a hardware failure you will have to contact technical support.

2.1.6 Testing Daughtercard Connectors

This test requires a test fixture and cannot be performed by the user.

3 Reference Design Types

"The Reference Design" in this chapter refers to the FPGA designs located on the user CD at
D:\FPGA_Reference_Designs\DN9200K10PCIE8T\MainRef\
D:\FPGA_Reference_Designs\Programming_Files\DN9200K10PCIE8T\MainRef\

Four other self-contained designs are on the CD and described in this manual. These four designs are described in their own sections later in this chapter. The remaining sections describe the "MainRef" design. "MainTest", "The reference design" and "The Dini Group reference design" are the same thing.

The four additional designs are
PCIe Interface Design: Tests the 64-bit interface between FPGA A and the LX50T (PCIe)
LVDS Reference Design: Characterizes the FPGA interconnect using source-synchronous
Ethernet Reference Design: Tests the Ethernet PHY.

Other features of the board, such as memory sockets and daughtercard headers are tested using the Main Test.

3.1 Main Test

This reference design is also referred to as "SINGLE INTERCON", because it is used to test the FPGA-to-FPGA interconnect. This reference design provides access to the following:

-All FPGA clocks
-DDR2 memory
-MainBus (for USB and PCI Express)
-RS232
-"Tenth Inch" header pins

3.2 LVDS

This reference design is an implementation of Xilinx App Note 705. It achieves 900 Mt/sec per LVDS pair between FPGAs, the maximum speed possible using this method. (Other methods may improve bandwidth beyond this limit). The design provides MainBus registers to allow counting the bit error rate of each bank of 40 interconnect pins.

3.3 Single Fast

This reference design allows the characterization of FPGA-to-FPGA interconnect using standard synchronous IO methods between FPGAs. Main Bus registers are provided to allow the monitoring of the BER of each bank of 40 interconnect pins.

3.4 V5 Interconnect

This reference design might not be provided.

3.5 Ethernet

This reference design is a hardware test of the Ethernet interface. It may not be provided.

3.6 Header

This reference design is a hardware test of the Header interface. It requires a test fixture to work properly. It may not be provided.

4 Using the Reference Design

4.1 Reference Design Memory Map

Each reference design uses the MainBus interface to supply status and controls. The following memory map is used. These registers are accessible using the windows USB Controller program using the "MainBus" menu, or from AETEST for PCI Express access.

All addresses on main bus are 32-bits. Each address contains one 32-bit word. By convention, each FPGA has a fixed memory range. FPGA A will respond to all MB accesses in the range 0x00000000 – 0x0FFFFFFF. FPGA B will respond to accesses from 0x10000000-0x1FFFFFFF. Other addresses are not defined.

The addresses given below are offsets from the base address of any given FPGA. Some registers are not valid for all FPGAs. Some addresses are not valid for all of the Dini Group's reference designs. (Main Test does not have LVDS registers, and LVDS test does not have DDR2 registers).

Some of the address bits are decoded as "Don't care" bits. Therefore, accesses to undefined addresses may alter stuff.

Address Range	Register Name	Register Contents
0x00000000 - 0x07FFFFFF	DDR2	Mapped to the DDR2 SODIMM memory
0x08000001	DDR2HIADDR	Upper bits of DDR2 address (MainBus memory space is smaller than most DDR2 SODIMMs)
0x08000002	IDCODE	0x05000142
0x08000003	DDR2HIADDRSIZE	The number of valid addresses in DDR2HIADDR
0x08000004	INTERCONTYPE	An ID code used to identify which design is loaded 0x34561111 – Interconnect, Single 0x34562222 – Interconnect, LVDS 0x34563333 – Interconnect, LVDS (reversed) 0x34560000 – Any Other Design (PCIe, Ethernet, etc.)
0x08000005	DDR2SIZE	A code to control how DDR2 memory is coded into MainBus memory
0x08000006	RWREG	Read/Write Scratch Register for testing
0x08000007	DDR2TAPCNT0	The current "tap" settings of the IODELAY elements in the DQ IO buffers on the DDR2 interface (lower bytes)
0x08000008	DDR2TAPCNT1	The current "tap" settings of the IODELAY elements in the DQ IO buffers on the DDR2 interface (upper bytes)
0x0800000A - 0x080000011		This range of addresses is reserved for manufacturing tests (Daughtercards)
0x080000012	SODIMM_SEL	Does nothing on the DN9200K10PCIE8T
0x080000013	FAN_TACH	The current input value of the fan tachometer (0 or 1)
0x080000014	IS_LX_330	0x1 if the FPGA is an LX330, 0x0 is it is not.
0x08000001B	SODIMM_RANK	Data read from the SODIMM IIC interface
0x08000001C	SODIMM_COL	-
0x08000001D	SODIMM_ROW	-
0x08000001E	SODIMM_BANK	-
0x08000001F	SODIMM_CAS	-
0x08000021	CLK_COUNTER	Contains contents of G0 counter /4
0x08000022	CLK_COUNTER	Contains contents of G1 counter
0x08000023	CLK_COUNTER	Contains contents of G2 counter
0x08000024	CLK_COUNTER	Contains contents of CLK48 counter
0x08000025 - 0x08000032	RCLK_COUNTER	LVDS source-synchronous clock counters (LVDS design only)
0x08000033 - 0x0800003F	MCLK_COUNTER	Clock counters for (in backwards order!): DDR2 clock, EXTCLK0, EXTCLK1, SMACLK, CLK_FBE, CLK_FBB, CLK125_ETH, CLKP, CLK_TPp
0x08000040 - 0x08000043	DDR2TESTTAPCNT	Reserved for manufacturing tests (DDR2)

0x08000044	LED_OE	Controls LED output enables.
0x08000045	LED_OUT	Controls LED output values.
0x08000046	DDR2SIZE_SODIMM2	Controls address mapping order on second DIMM interface (FGPA C only)
0x08000047	HIADDRSIZE_SODIMM2	Number of unique addresses in HIADDR for second DIMM interface (FPGA C only)
0x0800004B	SODIMM2_RANK	IIC data retrieved from the SODIMM in socket 2 (FPGA C only)
0x0800004C	SODIMM2_COL	
0x0800004D	SODIMM2_ROW	-
0x0800004E	SODIMM2_BANK	-
0x0800004F	SODIMM2_CAS	-
0x0800007E	VRP_ALL	Contains input signals on the "VRP" pins
0x0800007F	VRN_ALL	Contains input values on the "VRN" pins
0x0B000000 - 0x0B0003FF	BLOCKRAM	Contents of an internal-FPGA block RAM
0x0C000XX0	BUS XX OUT	XX can be 0-21 hex. Current output status of IOs on bus XX.
0x0C000XX4	BUS XX OE	XX can be 0-21 hex. OE status of IOs
0x0C000XX8	BUS XX IN	XX can be 0-21 hex. The input values
0x0C000XXC	BUS XX Name	A unique name of the bus (schematic)
0x0xxxxxxx	REG_DEFAULT	0xDEAD5566 Any undefined register

5 Interconnect (Single)

The "single-ended" interconnect test tests the DC connectivity of FPGA-to-FPGA interconnect, and the "MB" signals.

Presented on the MainBus, are registers allowing the interface to control the output value, output enable, and input value of each FPGA-to-FPGA interconnect pin. Each pin on the FPGAs is pulled high. This allows a test program to find single-stuck-at faults, open faults, and stuck-together faults.

5.1 Using the Design

The design can be controller over the MainBus. The register banks connected to the IO are arranged into "busses". Each bus has an ID code, an OE register bank, an ENABLE register bank, and an IN register bank.

The addresses of the IO registers are as follows:

FpgaNum (4-bit) | MB_SEL_INTERCON (4 bit) | busnum (20-bit) | reg_offset (4-bit)

FPGA NUM is 0x0 for FPGA A, 0x1 for FPGA B, 0x2 for FPGA C…
MB_SEL_INTERCON is 0xC
busnum is any number, but only low-values (less than LAST_ADDR) will constrain valid busses
reg_offset is 0x0 for REG_OUT, 0x4 for REG_OE, 0x8 for REG_IN, and 0xC for REG_ENABLED

To determine which bits (if any) in a bus are valid, read the REG_ENABLED register. The 32-bits returned '1 are a mask for which of the bits in the REG_OUT, REG_OE, and REG_IN registers are meaningful.

To get the bus ID of a bus, write value 0x1 (32-bit) to REG_ENABLED, then read REG_ENABLED, then write 0x0 (32-bit) to REG_ENABLED. The value returned will be a coded name for the bus. Bits 0-15 are ASCII characters representing FPGA names. Bits 16-31 are an arbitrary unique integer distinguishing the bus. Connecting busses from two different FPGAs have the same bus ID.

To cause an FPGA to output signals on a bus, write 0xFFFFFFFF on REG_OE. To set the outputs all to "high" write 0xFFFFFFFF to REG_OUT.

To read the current received value from the bus' inputs, read from REG_IN

5.2 Running the Test

In the USB Controller program, select Settings->OneShot Test. From the dialog box, check the Interconnect Test box. The program will automatically load the bit files, set the clocks and run the test.

5.3 DDR2 Interface

The DDR2 interface design is an example DDR2 controller running at 250MHz. You can use this controller as an example, especially for the purpose of required IO logic, timing and clocking. The controller bandwidth is most of the DDR2 bandwidth possible on the DN9200K10PCIE8T.

5.4 Provided Files

The DDR2 reference design is part of the "MainRef" reference design, and the MainRef files should be used.

5.5 Using the Design

The DDR2 memory interfaces are mapped to the address range

0xNXX00000 – 0xNXXFFFFF

Where the 4-bit "N" represents an FPGA ID, as described in the MainBus interface description. X are "don't-care". Since the remaining 19 bits are insufficient to address an entire 4GB DRAM, there is a register DDR2HIADDR that selects the highest address bits of the DRAM. Each

address refers to a 32-bit location in the DRAM. The lowest bit is not mapped to DRAM address, but instead selects between the upper and lower 32 bits of the DRAM data. This is necessary because MainBus is a 32-bit interface, and the DN9200K10PCIE8T DRAM interfaces are 64 bits wide.

The bank and side controls are also mapped to the DDR2HIADDR register. The location of the DDR2HIADDR register is given in the Reference Design Memory Map section.

The clock that this design uses (G1) must be set to between 180 and 250MHz. <verify this number>.

5.6 Running the Test

To run the hardware test, in the USB Controller application, select Settings->OneShotTest and check the DDR2 box. The program will automatically load the bit files, set the clocks and run the test, reporting any errors.

5.7 Clock Counters

Each clock available to the FPGA is connected to a counter register, and the value of this register is available on MainBus. In this way, the user can determine if each clock input is working properly.

5.8 LEDs

All of the LEDs are connected to an output enable register. When the LEDs are not enabled, the blink a pattern representing which FPGA the design is for. When enabled, each LED is controlled by the LED value register.

5.9 Simulating the Reference Design

The simulation environment the Dini Group uses is ModelSim. A ModelSim project file is provided, but it may not be compatible with your version of ModelSim. When you create a ModelSim project, add only the top-level design file (sim_board.v).

Source can be found on the user CD:
D:\ FPGA_Reference_Designs\DN9200K10PCIE8T\MainRef\source\

Also, you must add to the project a simulation library. Simulation models of all of the primitives used in the reference design are found in the Xilinx ISE install directory in the unisims directory.

Simulation models are also provided of the DN9200K10PCIE8T as a whole board, along with DDR2 modules, headers and the MainBus interface.

6 LVDS Reference Design

The "LVDS Interconnect" design is to show the user how to implement source-synchronous communication between FPGAs. Using this method, the advertised 900Mbs system speed can be achieved. If you do not wish to use source-synchronous interconnect, ignore this reference design with prejudice.

All FPGA-to-FPGA interconnect in this design is constantly being driven by one FPGA sending (uni-directionally) a test pattern. The receiving FPGA checks the test pattern for correctness against a known pattern.

The design is intended to characterize the bandwidth of the interconnect between FPGAs. Access to test status is provided over the MainBus interface.

Note that there are two designs, "ADC" and "CBA". In the design, the directions of LVDS connections between FPGAs are uni-directional. In the "CBA", all of the signals are in a direction opposite to the "ABC" design signals.

6.1 Provided Files

The source is located at:

D:\FPGA_Reference_Designs\DN9200K10PCIE8T\MainRef

Note that this is the same source as the "Main Reference Design". To compile the design for LVDS, `#define` statements in the Verilog code must be added or removed. The make.bat utility described in the "compiling the reference design" section automatically adds and removes these directives. The pre-compiled bitfiles for this design are located at

D:\FPGA_Reference_Designs\Programming_Files\DN9200K10PCIE8T\LVDSIntercon\

6.2 Using the Design

The design's MainBus interface is undocumented

The IOs in the LVDS reference design are clocked using the G0 clock. A clock setting of 300 MHz on G0 results in data transmission from FPGA to FPGA of 600 Mbs per signal pair.

The G2 clock is required to be 200 MHz, or IDELAY will not calibrate correctly, and performance will be degraded.

6.3 Running the Test

In the USB Controller program, select Settings->OneShot Test. From the dialog box, check the Interconnect Test box. The program will automatically load the bit files, set the clocks and run the test.

6.4 Implementation Details

Mostly, the LVDS design follows the Xilinx application note

6.4.1 Lane Alignment

The Xilinx application note only allows for the bit alignments so that all bits on a 16-bit bus are output as 8-bit words in the slow clock domain on the receiver FPGA. However, it's important to note that the alignment of the 8-bit words may be off by one cycle. That is, the cycle latency from one FPGA to another may be different from one byte lane to another. Additionally, the latency might change each time the bit alignment machine retrains. If you wanted to fix this you would have to put in some sort of automatic cycle delay element.

6.4.2 Funny Banks

Not all banks on the Virtex-5 FPGA have a BUFR resource available. In order to implement the LVDS design, we had to swap out the BUFR for a dynamically-adjusted clock from a DCM.

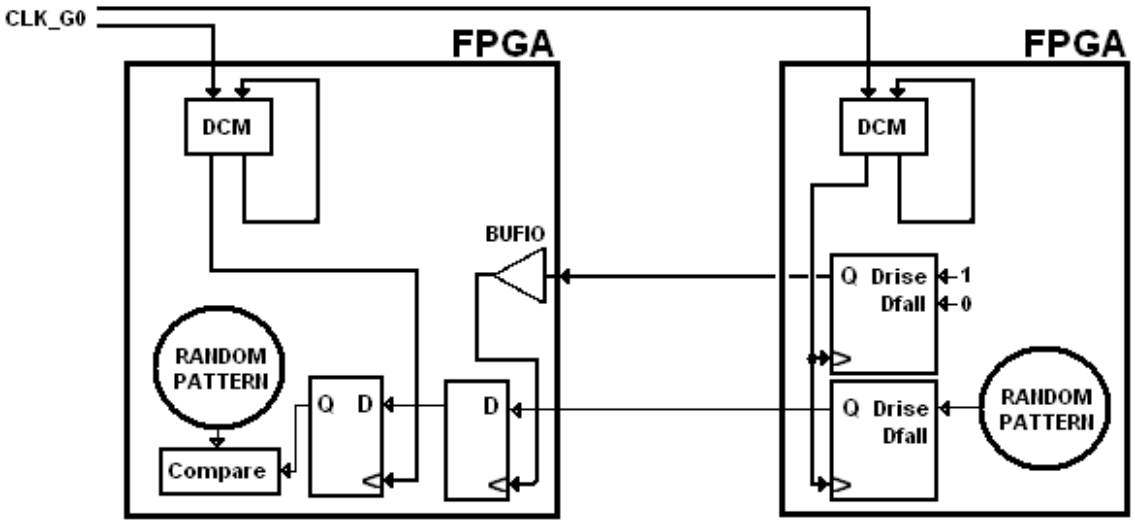


Figure 136 - LVDS Reference design clocking global

Here is how the design is supposed to look, according to the app note.

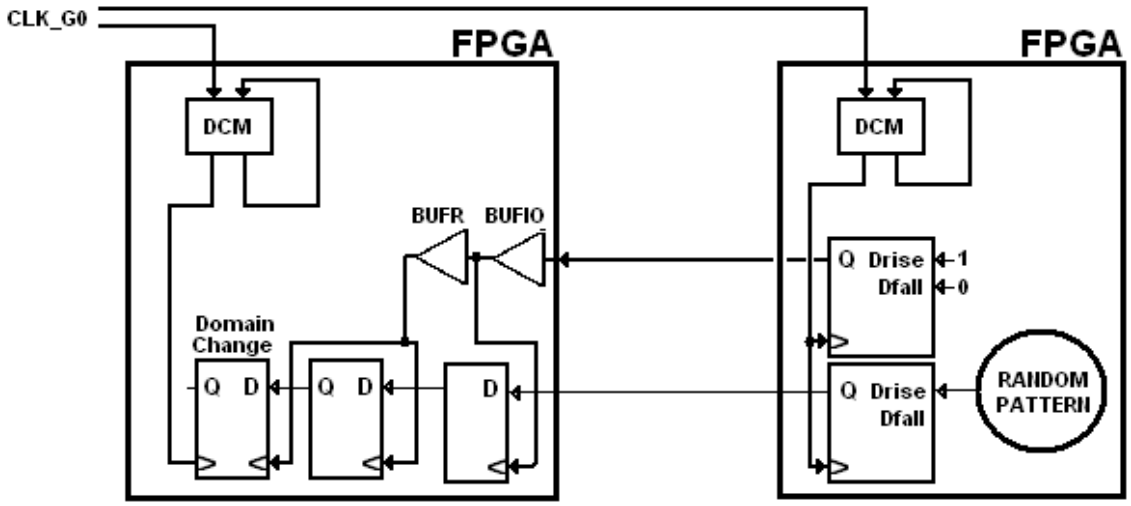


Figure 137 - LVDS Reference design clocking local

There is no difference in performance between the two methods, because the clock in question is not part of the critical data path. (The BUFIO).

7 PCIe Interface Reference Design

The PCIe reference design is an example of how to use the provided pcie_x8_user_interface.v module provided.

7.1 Provided Files

D:\FPGA_Reference_Designs\common\PCIE_x8_Interface

7.2 Using the Design

The PCIe reference design maps internal FPGA block rams to BAR 1 through BAR6 of the FPGA's PCIe interface, and a separate block ram to the DMA channel of the PCIe interface. When the design in loaded in the FPGA, a host machine can read and write to this memory space to verify the interface is working. Only 4 kB of memory is mapped to each BAR, even though the size of each BAR is larger. The block ram memory will wrap.

7.3 Running the Test

The PCIe Reference Design is an FPGA A-only design that implements the pcie_x8_user_interface module described the document

D:\FPGA_Reference_Designs\common\PCIE_x8_interface\pcie8t_user_interface_maual.pdf

This design implements a PCIe target access and DMA interface to a block ram inside FPGA A. The source code is located on the CD at:

D:\FPGA_Reference_Designs\Programming_Files\pcie_fpga\pcie_dma

The pre-compile bitfiles for your board are located at:

D:\FPGA_Reference_Designs\Programming_Files\pcie_fpga\pcie_dma

In this design, accesses to BAR2, BAR3, BAR4, BAR5 and both DMA channels are mapped to separate block rams in the FPGA. Upper bits of the address offset are ignored, so the block ram loops around. To use this design, see the PCIe section of the hardware chapter.

1 Compiling the Reference Design

All source code for the reference design is included on the CD and may be used freely by customers for anything legal. The MainRef reference design can be found on the user CD here.

D:\FPGA_Reference_Designs\
\common\DDR2\controller_ver*
\common\DDR2\ddr2_to_mb*
\DN9200K10PCIE8T
\MainRef\source*

The top module is

D:\FPGA_Reference_Designs\DN9200K10PCIE8T\MainRef\source\fpga.v

This module includes all of the other required sources and expects the directory structure found on the CD.

1.1 The Xilinx Embedded Development Kit (EDK)

The DN9200K10PCIE8T does not use the EDK because it has no embedded processor.

1.2 Xilinx ISE

Xilinx ISE version 10.1 (service pack 1 or later) is required to use the reference designs. Earlier versions may work, but are not supported.

If you are using a third-party synthesis tool, you can create a new ISE project file and add the .edf as a source. For part type, select the type of FPGA installed on your board. Make sure to add the provided .ucf file to the project, or the produced place-and-route will not work.

Run the map, implement and generate steps.

1.3 The Build Utility: Make.bat

If you are not using a third-party synthesis tool, then you should use the provided batch script to generate the programming files from the reference design. The batch script will synthesize using XST from the source, assigning the correct value to each `#define` switch in the source.

The Build Utility is found at 'DN9200K10PCIE8T/build_xst/make.bat'. This batch file can be used to run XST, ISE and bitgen. You may need to run make.bat from inside of a Cygwin session, or otherwise have the program `sed` installed. You may also need to add the Xilinx bin directory to your path so the command "par" calls the correct program.

There are command line options that cause the script to output the correct reference design. (Since all the reference designs use the same source files). Most commonly, you would want to make the "single-ended" or "main" reference design. This includes the DDR2 controller.

Type
`>make.bat SINGLE`
to change the current source compilation type to "Single ended". Then type
`>make.bat LX330`
to change the current place-and-route type to LX330. Then type
`>make.bat`
to start synthesis, place and route, and bitfile generation. The build script creates a directory called "out" and places its output files there. After the script completes you will find files for each FPGA that was built. fpga_*.bit is the file to be downloaded to the FPGA.

When using the provided VHDL, the generic definitions are not complete in the Dini Group code. Some of the signals that are governed by generics must be defined externally or (defined in the first place).

1.4 Bitgen Options

The Make.bat script correctly sets all bitgen options that are compatible with the DN9200K10PCI. The following options should be used with the DN9200K10PCI. Options that are not listed here can be selected by the user, or left to their default settings.

Compress:	OFF	(Or you can disable "sanity check" option on board)
UnusedPin:	Pullnone	
Persist:	Yes	(Only required if Readback is used)
Encrypt:	No	(YES requires that you disable "sanity check" option on board)
DonePipe:	No	
DriveDone:	Yes	

Don't ever disable "CRC Check". This is the easiest and most certain way to turn your FPGAs into little piles of carbon ash. I am pretty sure this option exists to increase sales of replacement FPGAs.

1.5 VHDL

The VHDL version of the reference design is included along with the Verilog version. The VHDL is a translation of the Verilog. It's updates are less granular and lag by a few months. It also may contain translation bugs that we haven't noticed. All of the pre-compiled bit files are generated from the Verilog source. If at all possible I would go with the Verilog. The reference design gets undocumented minor updates on a weekly basis. If you need a specific update, we can re-generate and test the VHDL for you.

Chapter 6: Ordering Information

Part Number
DN9200K10PCIE8T

1 How to order

Request quotes by emailing sales@dinigroup.com.

Fax a PO to: (858) 454-1728
Do not fax cash.

For technical questions email support@dinigroup.com

2 Optional Equipment

The following tools are suggested for use with the Dini Group DN9200K10PCIE8T.

2.1 Compatible Dini Group products

The Dini Group supplies standard daughtercards and memory modules that you can use with the DN9200K10PCIE8T.

2.1.1 Interface Boards

Debugging Connections

Mictor
http://dinigroup.com/dnsodm200_mictor.php
http://dinigroup.com/dnsodm200_quadmic.php

2mm Header
http://dinigroup.com/dnsodm200_intercon.php

PCI (3.3V)
(Contact Us)

USB (Host, peripheral, or OTG)
http://dinigroup.com/dnsodm200_usb.php

2.1.2 Memories

The memory module solutions from Dini Group allow the user to install whichever type of memory his application requires.

SRAM (Synchronous) 64 x 1Mb @175 MHz
GSI part number GS8320V32
DNSODM200_SRAM
http://dinigroup.com/dnsodm200_ssram.php

Zero Bus Latency SRAM
(Contact Us)

RLDRAM 64 x 1Mb x 8bank
Micron part number MT49H8M32
DNSODM200_RLDRAM
http://dinigroup.com/dnsodm200_rldram.php

DDR3 64 x 16Mb @ 250 MHz
http://dinigroup.com/dnsodm200_ddr3.php

DDR1 64 x 32Mb @ 175 MHz
http://dinigroup.com/dnsodm200_ddr1.php

DRAM (Synch) 64 x 16Mb @75 MHz
http://dinigroup.com/dnsodm200_sdr.php

Mobile SDRAM
Micron MT48H32M16
http://dinigroup.com/dnsodm200_se.php

NAND Flash
Intel StrataFlash PE28F256P30
http://dinigroup.com/dnsodm200_se.php

NOR Flash 64 x 8Mb @ 66 MHz
Spansion S71WS128NB0BFWAN0
http://dinigroup.com/dnsodm200_flash.php

PSRAM 32 x 4Mb @ 66 MHz
Spansion S71WS128NB0BFWAN0
http://dinigroup.com/dnsodm200_flash.php

2.1.3 Daughter cards

Dini Group daughtercards connect to the MEG-Array connector (400-pin) using the standard Dini Group daughter card interface description.

PCI Express
8 lanes
http://dinigroup.com/dnmeg_v5tpcie.php

FPGA-to-FPGA Interconnect
Connect two adjacent daughtercard connectors
http://dinigroup.com/DNMEG_Intercon.php

Board-to-Board Interconnect
http://dinigroup.com/DNMEG_Mictor_Diff.php

0.1" Header
http://dinigroup.com/DNMEG_Obs.php

DVI and HDMI
http://dinigroup.com/dvidc.php

High-Speed Serial (10Gig Ethernet, HSSDC, SATA, FibreChannel, XAUI)
http://dinigroup.com/dnmeg_v5t.php

ADC and DAC
11+ ENOB @ 210 MHz
http://dinigroup.com/DNMEG_ADDA.php

Mictor
http://dinigroup.com/DNMEG_Mictor_Diff.php

Riser Card

Dini Group T-Shirts, Hats
FPGA Mood-rings

2.2 Compatible third-party Software

PCI Tree
http://www.pcitree.de/

CatScan
http://www.getcatalyst.com/catalystcatscan.html

Putty
http://www.chiark.greenend.org.uk/~sgtatham/putty/

2.3 Compatible third-party hardware

The following products are recommended for use with the DN9200K10PCIE8T

Standard DDR2 SODIMM modules
www.crucial.com
4GB - $550

2GB - $54
1GB - $21
512MB - $10

Xilinx Platform II USB Cable
HW-USB-II-G
http://nuhorizons.com
(required for JTAG connection to FPGA, ChipScope)

Mictor breakout, Mictor Cables
MIC-38-BREAKOUT, MIC-38-CABLE-MM-18
http://www.emulation.com/catalog/off-the-shelf_solutions/mictor/

PCI Express riser card
PEX16LX $120
http://www.adexelec.com/pciexp.htm

PCI Express 2.0 Motherboard
Asus P5E WS PRO LGA 775 Intel X38 ATX Server Motherboard
http://www.newegg.com/

3 Compliance Data

3.1 Disclaimer

Information is the manual is "as is" something about liability and medical devices, and space exploration.

Figure 138 - Disclaimer block diagram

Reference design and software might not work. Don't put all your money in only one or two stocks, etc.

3.2 Compliance

3.2.1 FCC EMI

Since the DN9200K10PCIE8T is not intended for production systems, it has not undergone EMI testing. An FCC Compliance Screening can be done by special request, but requires the customer to provide a sample end use system with good EMI shielding.

3.2.2 PCIe-SIG

The DN9200K10PCIE8T passes the electrical compliance test for PCI express 1.1 and 1.0a, using the Provided DMA-enabled PCI Express core, and with the Xilinx PCIe endpoint LogiCORE. Additionally, the LogiCORE endpoint passes the PCI-SIG compliance full test.

The provided PCI Express DMA-enabled core has not been tested at a compliance workshop.

The FX70T passes the PCI Express electrical compliance test for revision 2.0.

EYE WIDTH:	149ps
TIE JITTER:	-28 to 28ps
TOTAL JITTER @ BER:	77ps
DIFF PEAK VOLTAGE	1.12V

3.3 Environmental

3.3.1 Temperature

The DN9200K10PCIE8T is designed to operate within an ambient temperature range of 0 – 50 °C.

In environments with a high ambient temperature, or where the total heat capacity of the adjacent air flow is restricted (such as inside a server), a new thermal evaluation will be required. All components on the DN9200K10PCIE8T are rated to operate within a temperate range of 0° to 80°C.

Dini Group has some larger Heatsinks and Fans if you need another few C° of temperature headroom.

3.4 Export Control

3.4.1 Lead-Free

The DN9200K10PCIE8T meets the requirements of EU Directive 2002/95/EC, "RoHS". Specifically, the DN9200K10PCIE8T contains no homogeneous materials that:

a) contains lead (Pb) in excess of 0.1 weight-% (1000 ppm)
b) contains mercury (Hg) in excess of 0.1 weight-% (1000 ppm)
c) contains hexavalent chromium (Cr VI) in excess of 0.1 weight-% (1000 ppm)
d) contains polybrominated biphenyls (PBB) or polybrominated dimethyl ethers (PBDE) in excess of 0.1 weight-% (1000 ppm)
e) contains cadmium (Cd) in excess of 0.01 weight-% (100 ppm)

No exemptions are claimed for this product.

3.4.2 The USA Schedule B number based on the HTS

8471 60 7080

3.4.3 Export control classification number ECCN

EAR99